

Declarative Policy-based Adaptive Mobile Ad Hoc Networking

Changbin Liu* Ricardo Correa* Xiaozhou Li* Prithwish Basu† Boon Thau Loo* Yun Mao‡

*University of Pennsylvania †Raytheon BBN Technologies ‡AT&T Labs - Research

{changbl, ricm, xiaozhou, boonloo}@seas.upenn.edu, pbasu@bbn.com, maoy@research.att.com

Abstract—This paper presents DAWN, a declarative platform that creates highly adaptive policy-based MANET protocols. DAWN leverages declarative networking techniques to achieve extensible routing and forwarding using declarative languages. We make the following contributions. First, we demonstrate that traditional MANET protocols can be expressed in a concise fashion as declarative networks and policy-driven adaptation can be specified in the same language to dictate the dynamic selection of different protocols based on various network and traffic conditions. Second, we propose inter-protocol forwarding techniques that ensure packets are able to seamlessly traverse across clusters of nodes running different protocols selected based on their respective policies. Third, we have developed a full-fledged implementation of DAWN using the RapidNet declarative networking system. We experimentally validate a variety of policy-based adaptive MANETs in various dynamic settings using a combination of ns-3 simulations and deployment on the ORBIT testbed. Our experimental results demonstrate that hybrid protocols developed using DAWN outperform traditional MANET routing protocols, and are able to flexibly and dynamically adapt their routing mechanisms to achieve a good tradeoff between bandwidth utilization and route quality. We further demonstrate DAWN’s capabilities to achieve inter-protocol forwarding across different protocols.

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 [13] and AODV [25] set up routing state *on demand* and hence are preferred for low traffic environments; proactive routing protocols such as OLSR [9] and HSLS [30], on the other hand, expend network bandwidth to gather network topology state with the purpose of amortizing this extra cost over multiple traffic flows – hence these are in general better for environments with greater traffic loads and more number of source-destination pairs. Recently researchers have focused on routing in MANETs that are at best intermittently connected (a class of disruption tolerant networks or DTNs) – examples are epidemic routing protocols [33], [28] and probabilistic/predictive routing protocols [15].

Due to a wide range of variability in network connectivity and mobility, and also a wide range of data traffic patterns, we argue that 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. [11], [29], [26], [12], [36], [24]) have systematic logic behind their design, they are not very flexible and customizable as they are specified

in a stove-piped fashion. As a result, the *no-one-size-fits-all* argument applies to these hybrid protocols as well. 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.

To address the above challenges, we present *Declarative Adaptive Wireless Networking* (DAWN), a platform that creates highly customizable hybrid protocols by *composing* any number of known protocols, and utilizes a declarative policy-based framework to define the rules and conditions for switching among different protocols. DAWN achieves these capabilities as follows. First, known protocols such as the ones in the “link-state” and “epidemic” families are specified in a database query-style declarative language. Second, rule-based adaptation policies dictating when to use which protocols and under what conditions are specified in the same language. Finally, the runtime system automatically compiles the protocols and policies into actual implementations.

DAWN has the following benefits: (1) hybrid protocols written in a 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) DAWN 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 smaller in size than the corresponding imperative implementations in languages like C/C++. Furthermore, shared protocol components can be reused and composed to create new hybrid protocols. Specifically, our contributions include:

Policy-based adaptive MANET routing: DAWN provides a unified platform based on declarative networking [6], [19] that enables one to implement a variety of MANET routing protocols (proactive, reactive, and DTN) concisely in a few lines of code. Moreover, policy-based decisions for creating hybrid protocols can be expressed in the same declarative language, and used to switch between different protocols as the network conditions (e.g., connectivity, mobility and traffic volume and patterns) change over time.

Policy-based inter-protocol forwarding: DAWN’s hybrid approach results in different protocols being executed at different nodes in the network. For instance, some disconnected nodes in the network may utilize DTN routing, while other *better connected* nodes are running reactive or proactive protocols. We propose inter-protocol forwarding techniques that ensure packets are able to seamlessly traverse across clusters of nodes running different protocols. Interestingly, these forwarding

policies can themselves be customized using additional forwarding policy rules, and applied to packets as they traverse through different protocols.

Experimental evaluation: We have developed a full-fledged implementation of DAWN using the RapidNet declarative networking system [4]. We experimentally validate a variety of policy-based adaptive MANET routing protocols in various dynamic settings. Using a combination of ns-3 [1] simulations and deployment on the ORBIT [2] testbed, we evaluate these protocols under varying mobility, connectivity, and traffic patterns, and demonstrate that they can outperform traditional MANET routing protocols – in particular, they are able to flexibly and dynamically adapt their routing mechanisms to achieve a good tradeoff between bandwidth utilization and route quality. We further demonstrate DAWN’s capabilities to forward packets in a network consisting of clusters of nodes running proactive, reactive and DTN routing protocols.

To the best of our 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 contributions on policy-based adaptive MANET routing, our work also demonstrates that declarative networking techniques can be used effectively to rapidly prototype, deploy, and compare a variety of MANET protocols. Moreover, the proposed declarative framework facilitates the ability to rapidly explore a wide range of deployment and implementation parameters necessary for tuning the performance of MANET protocols.

II. OVERVIEW

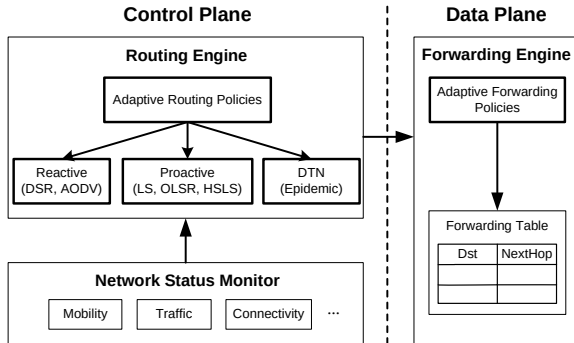


Fig. 1. Components of a DAWN node.

Figure 1 shows an overview of DAWN from the perspective of a single DAWN node’s network layer, divided into control plane (left) and data plane (right). DAWN is a declarative platform for network designers to: (1) develop new MANET protocols; (2) customize policies for adaptive routing and forwarding; and (3) evaluate prototypes in simulation and implementation modes. DAWN is deployed in a fully distributed fashion. Each node runs a DAWN instance, and communicates with other nodes by executing the input declarative protocols and policies (shown as bold boxes in the figure).

A. DAWN Components

In the *routing engine* component, a variety of routing protocols, ranging from proactive (e.g. Link-State, OLSR, HSLs [30]), reactive (e.g. DSR, AODV [25]) and DTN (e.g. epidemic routing) are implemented as declarative protocols.

Policy-based routing adaptation is dictated by *adaptive routing policies* via a series of adaptation policy rules [16] specified in the same declarative language as routing protocols. These *adaptive routing policies* are usually composed by network designers according to application demands and network scenarios, and are highly customizable. Example adaptation policies include switching between Link-State (LS) and HSLs due to network mobility for the tradeoff between route quality and communication overhead, or between epidemic routing when the network is highly disconnected and proactive routing when connectivity is reestablished.

As input to these policy rules, a *network status monitor* is deployed to continuously monitor network status (e.g. network mobility, traffic and connectivity information). The monitoring queries themselves are also specified as declarative rules.

Depending on the type of protocol, the output of executing these declarative protocols is typically forwarding tables in the data plane for use in determining the next hop to forward a packet given a destination, or in the case of source routing protocols, the entire path required to traverse from the current node to the destination. A *forwarding engine* within data plane interacts with routing engine for forwarding data traffic. Interestingly, these forwarding policies can themselves be customized using additional forwarding rules, and applied to packets as they traverse through different protocols.

B. Background

Given our planned use of declarative networking as a framework for policy-based MANETs and associated adaptation policies, we begin with a brief overview of its capabilities and prior use cases. 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.

NDlog enables a variety of routing protocols and overlay networks to be specified in a natural and concise manner. For example, traditional routing protocols such as the path vector and distance-vector protocols can be expressed in a few lines of code [6], and the Chord distributed hash table in 47 lines of code [19]. When compiled and executed, these declarative protocols perform efficiently relative to imperative implementations. As evidence of its widespread applicability, declarative techniques have been used in several domains, including, fault tolerance protocols [31], cloud computing [5], sensor networks [8], overlay network compositions [20], wireless channel selection [17], and as a basis for course projects in a networked systems class [10].

In addition to ease of implementation, another advantage of the declarative networking approach is its amenability to formal and structured forms of correctness checks. These include the use of theorem proving [34], algebraic techniques for constructing safe routing protocols [35], and runtime verification [38]. These formal analysis techniques are strengthened by recent work on formally proving correct operational semantics of *NDlog* [23]. Finally, the dataflow framework used in declarative networking naturally captures information flow as distributed queries, hence providing a natural way to use the

concept of *network provenance* [37] to analyze and explain the existence of any network state.

NDlog is based on Datalog [27]: a Datalog program consists of a set of declarative *rules*. Each rule has the form $p :- q_1, q_2, \dots, q_n$, which can be read informally as “ q_1 and q_2 and $\dots$ and q_n implies p ”. Here, p is the *head* of the rule, and $q_1, q_2, \dots, q_n$ 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 ($\langle \rangle$). 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.

In DAWN, each node runs its own set of *NDlog* rules. Typically, these rules are common across all nodes (e.g. all nodes run the same protocol), but may further include per-node policy customizations. *NDlog* rules 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 [14], 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, and selections) 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 routing protocols can be expressed concisely using the declarative networking framework. We present typical protocols of each category in proactive, reactive and DTN-style MANET routing. This section sets the stage for Section IV where we discuss policies for routing adaptation.

A. Proactive Protocols

A well studied proactive protocol is the Link-State (LS) 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 updates (LSU) in traditional LS, followed by two variants of LS commonly used in MANET settings. Traditional dissemination of LSU 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 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).
```

In rule *ls1p* we utilize the *periodic* keyword to flood LSUs once in every 10 seconds. In order to ensure freshness of *lsu* tuples, they are stored using soft-state, where the lifetimes are set to be roughly the duration of periodic floods. In practice, a combination of triggered updates for timeliness and periodic updates for robustness are used. DAWN declarative platform 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 DAWN.

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-B, the following rules broadcast link information to all neighbors within the wireless range:

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

Once the entire network topology, i.e., all the links, are available at each node, additional rules are required in order to compute the shortest paths with minimum cost C for each source S and destination D. These rules take as input the local lsu tuples, and essentially result in the execution of the Dijkstra's algorithm locally. They are shown as follows:

```
bp1 path(@S,N,P,C) :- lsu(@S,S,N,C,W), P=f_init(S,N).
bp2 path(@S,D,P,C) :- lsu(@S,N,D,C1,W),
    bestPath(@S,N,P2,C2),
    C=C1+C2, P=f_concatPath(P2,D).
bp3 bestPathCost(@S,D,min<C>) :- path(@S,D,P,C).
bp4 bestPath(@S,D,P,C) :- bestPathCost(@S,D,C),
    path(@S,D,P,C).
```

In rule bp1, 1-hop paths are built from every link, while in rule bp2 paths are recursively constructed by concatenating shorter path with links. Rule bp3 computes the minimum cost for paths with same sources and destinations, and rule bp4 finally computes the best paths to all destinations originating from S. From the best paths, the following rule instantiates the forwarding table (in Figure 1) for each node S:

```
ft1 forwardingTable(@S,D,N) :- bestPath(@S,D,P,C),
    N=f_second(P).
```

Given that each bestPath tuple that stores the best path P from node S to D, rule ft1 uses the function f_second to extract out the next hop N along the path P to generate the corresponding forwardingTable entry.

Optimized Link-State Routing (OLSR): A well-known proactive MANET protocol is OLSR (Optimized Link-State Protocol) [9]. OLSR ensures efficient flooding by forwarding LSUs to a subset of neighbors known as multipoint relays (MPR). The union of the neighbor sets of MPRs of any node X is equal to the set of 2-hop neighbors of X. To implement OLSR-style flooding in LSUs requires modifications to rules ls1-2 to flood LSUs along MPR nodes. A detailed declarative OLSR rules and implementation can be found in [21].

Hazy-Sighted Link-State (HSLs): Hazy Sighted Link-State routing (HSLs) [30] 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 performs a 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.

HSLs rules 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,TTL-1) :- lsu(@Z,S,N,C,W,TTL),
    link(@Z,M,C1), TTL>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, HSLs rules can be modified to support triggered updates and batched triggered updates. In triggered updates, the flood of the corresponding LSU for each link insert/delete event is scoped using a similar HSLs policy, where LSUs flooded within a time interval has a fixed TTL. 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. Reactive Protocols

Next, we demonstrate a reactive protocol based on DSR [13]. The following set of rules show the route discovery of DSR (rules dsr1-4) followed by the route response (rules dsr5-7) traversing the best reverse path from destination to source.

```
dsr1 eRouteReq(@N,S,D,P,C) :- eQuery(@S,D),
    link(@S,N,C), P=f_init(S).
dsr2 eRouteReq(@Z,S,D,P,C) :-
    shortestRoute(@N,S,D,P1,C1), link(@N,Z,C2),
    C=C1+C2, P=f_concatPath(P1,N).
dsr3 minCost(@N,S,D,min<C>) :- routeReq(@N,S,D,P,C).
dsr4 shortestRoute(@N,S,D,P,C) :-
    minCost(@N,S,D,C), routeReq(@N,S,D,P,C).
dsr5 eRouteReply(@N,S,D,P,P,C) :-
    eRouteReq(@N,S,D,P,C), N=D.
dsr6 eRouteReply(@Z,S,D,P,P1,C) :-
    eRouteReply(@N,S,D,P,P2,C), Z=f_last(P2),
    f_size(P2)>0, P1=f_removeLast(P2).
dsr7 bestPath(@S,D,P,C) :-
    eRouteReply(@S,S,D,P,P2,C), f_size(P2)=0.
```

In DSR, a requesting node S issues an initial route request, denoted by eQuery(@S,D) event in rule dsr1. This results in a eRouteReq message tuple that is generated and recursively forwarded along all links (rule dsr2). The routeReq table is used to cache current route requests. To prune unnecessary paths, rules dsr3-4 ensures that only the shortest path from the initial node S to the intermediate node N is maintained.

Upon reaching the destination node D, rule dsr5 generates a eRouteReply message that is then sent back recursively via rule dsr6 along the computed best reverse path back to the requesting node S. The functions f_last and f_removeLast returns and removes the last node from a path respectively. Rule dsr7 generates the bestPath, based on the eRouteReply that has now traversed back to the original requesting node. The resulting bestPath can be used by node S to send a packet to D via source routing.

The rules for AODV [25] share similarities with DSR above, where only the next hop rather than the entire path is maintained.

C. DTN Protocols

As an example DTN-style protocol, epidemic routing has been proposed for reliable delivery in intermittently connected MANETs. A key reliability component of such protocols is the summary vector exchange as illustrated by 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 eMessage(@Y,I,S,D,P):- eBitVecReply(@X,Y,V),
    message(@X,I,S,D,P), f_vec_in(V,I)=true.
e4 message(@Y,I,S,D,P):- eMessage(@Y,I,S,D,P).
```

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 i th bit denotes the receipt of the i th 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 enable node X to filter (`f_vec_in`) local messages based on message ID I (sourced from S and destined to D with packet payload P) to be sent according to the bit vector V stored in the reply, which are then put in the local `message` table after transmission.

D. Evaluation of Declarative MANETs

We have performed a detailed evaluation of the above protocols in an emulated environment on our local testbed, and an actual deployment on the ORBIT testbed [2]. In our evaluation, DAWN instances are deployed on all nodes. Each declarative protocol is compiled and executed at all DAWN nodes, which execute the *NDlog* rules and communicate with other nodes to implement the distributed protocols. We briefly summarize the main findings of our evaluation.

To validate the correctness of our declarative MANET protocols, we compare their behavior and performance characteristics against prior observations of these protocols. Our results demonstrate that declarative MANET protocols can be efficiently implemented with performance characteristics that are expected from these protocols. For instance, we observe that the per-node communication overhead of LS 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 [16], [21], we have additional experimental details that demonstrate that the protocols are working correctly. For instance, LS and HSLS have periodic peak bandwidth utilization due to network-wide floods, and HSLS's peak bandwidth utilization is lower than LS. Furthermore, HSLS's flooding peaks vary over time based on current scope, and incur lower bandwidth utilization compared to OLSR.

In addition to LS, OLSR, and HSLS, our declarative implementations of DSR and summary-based epidemic routing also 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.

Category	Protocol	Rules	C++
Reactive	Dynamic source routing	11	659
	Traditional link state	15	728
Proactive	Optimized link state routing (OLSR)	34	1552
	Hazy sighted link state (HSLS)	18	956
Epidemic	Summary-vector based epidemic	17	755

TABLE I
DECLARATIVE MANET PROTOCOLS.

Table I summarizes the declarative protocols that we have successfully implemented, by comparing the corresponding number of rules required against the number of lines of code (LOC) in the generated C++ code. The generated code is a good estimation on the LOC required by a programmer to implement these protocols in a traditional imperative language. We note that the declarative specifications result in orders of magnitude reduction in code size.

IV. POLICY-BASED ADAPTIVE ROUTING

Building upon the basic declarative MANET protocols in the previous section, we next leverage DAWN's declarative framework to create customizable hybrid protocols by composing any number of known protocols. Specifically, DAWN will be used to synthesize policy-based adaptive *hybrid* routing protocols in its *routing engine* (Figure 1), i.e. protocols built from combining several existing protocols with specific criteria determining the usage of a particular protocol. These compositional capabilities are particularly useful for routing in heterogeneous network settings where features from various routing protocols (e.g. HSLS, OLSR, DSR, AODV, epidemic) are adaptively combined based on changing application demands and network conditions (connectivity, mobility, traffic flow characteristics, etc.).

To motivate policy-based adaptive hybrid protocols, we present some examples illustrating possible application scenarios. Our motivating examples are primarily rooted in military and emergency-response situations, where MANET is likely the most feasible form of communication in the absence of established wired infrastructures. In these scenarios, as personnel moves around, they may display varying mobility patterns that may not be temporally uniform. For instance, personnel may congregate together to discuss a mission (stable), and then dispersing out to execute mission operations (moderate to high mobility). There can also be periods of high communication or data transfer (e.g. during a VoIP call, or continuous stream of sensor and location feeds), followed by periods of low communication patterns (e.g. quiet operations in hostile territories). Mobility and traffic patterns may also have spatial aspects, where only a portion of the network is mobile, and different applications are deployed among clusters of nodes. Policy-based hybrid techniques for adaptation are well-suited for such temporally and spatially non-uniform environments.

To illustrate the plethora of possibilities, we first present an exemplar of a hybrid link-state protocol which adapts between two types of proactive protocols based on link dynamics and mobility, followed by a more generalized hybrid protocol that adapts based on traffic patterns, mobility and network connectivity.

A. Example: Hybrid Link-State Protocol

As the name suggests, a *hybrid link-state protocol* adapts between two variants of link-state routing protocols based on a single metric. While this example is a limited form of adaptation, its simplicity allows us to illustrate the salient features of our declarative rule-based adaptation approach.

The primary disadvantages of HSLs (Section III) is that it sacrifices optimality in routing to the need for scalability under high link dynamics. This is because unlike pure LS protocol which triggers updates when link status changes, HSLs forgoes the pursuit of gathering up-to-date information about the complete network topology and computes routes on a network topology that may have stale link state information. Imperfect topology knowledge may result in computation of suboptimal routes, and this effect can be announced in somewhat dynamic but sparsely connected topologies, where route diversity is limited.

There are a variety of metrics that one can use to quantify the degree of link-dynamics (due to mobility, duty cycling, outages, etc.) on a per-link basis in a network. The link metric we explore is *average availability* (AA) [28], which captures the average fraction of time that a link has been available for use in the recent past. Specifically, 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 locally computed based on gathered links, and is expressible using 5 declarative rules.

If the link AAs stop fluctuating wildly in most parts of the network as indicated by gathered links, one may decide to switch from HSLs to pure LS routing since that may yield near-optimal routes with a lower *stretch*¹ and higher validity. Moreover, a stable network will result in fewer triggered updates, ensuring that LS does not incur unnecessarily high control traffic.

Based on the LS and 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 neighbor links (or alternatively network-wide links based on collected LSUs). 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>) :- link(@M,N,C,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 gathered neighbor links. Note that we add one additional attribute which is link

¹Route *stretch* is the ratio of the hop count of the path between a source and a destination selected by the routing algorithm to that of the optimal path.

AA for every link tuple here. Rules s2-3 generate useHSLs and useLS predicates which are then added to rules hsls1-2 and ls1-2 respectively for protocol switching.

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 useHSLs and useLS without having to change the rules for the individual protocols themselves.

In addition, to dampen any effects on possible instability due to fluctuations of AA around the threshold value, one can modify the above rules such that protocol switching only occurs after the AA value has stabilized above or below the threshold value for a specified time period. After switching from one protocol to another, out-of-date routing tables and other information related to the previous protocol will be gradually phased out upon expiration.

B. Example: Generalized Hybrid Protocol

Our first example illustrates switching the underlying dissemination scheme² between two types of proactive protocols using only one metric alone. As an alternative example, in [16], we demonstrate that one can utilize the same declarative specifications above to achieve 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/dynamics, and (2) multi-path epidemic style message flooding in disrupted parts of the network under high mobility/dynamics.

Here, we present a more general version of an adaptive protocol, which we refer to as *generalized hybrid protocol* since it adapts across proactive, reactive, and DTN-style routing based on various network conditions. This generalized protocol is in fact a superset of the more constrained hybrid link-state and proactive-epidemic protocols mentioned earlier.

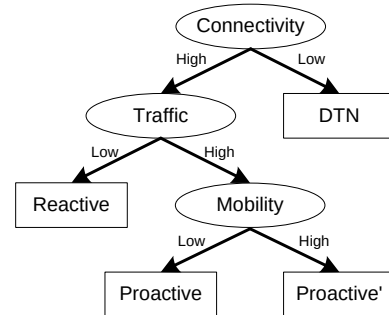


Fig. 2. An example decision tree that reflects one possible set of policies for adaptive routing. Ovals denote metrics and rectangles denote protocols.

Figure 2 illustrates this protocol in the form of a decision tree diagram. In addition to the AA metric for *Mobility*, this example uses two additional metrics gathered using the *network status monitor* in Figure 1: *Connectivity*, a measure of network

²Note that the route computation scheme at each node remains unmodified.

density computed based on a rule that uses a combination of “group-by”s and “aggregate”s at all nodes to determine the per-node degree (or degree within a neighborhood); *Traffic*, measured at each node using monitoring rules for data sending rate, packet queue length, etc.

Given the three metrics, the generalized hybrid protocol adapts as follows. In a highly disconnected environment (low connectivity), a DTN protocol (e.g. epidemic routing) is used. When network connectivity is well established, reactive routing is preferred if data traffic is low, since its route discovery procedure is only reactively data-driven and there is no overhead of maintaining up-to-date routing tables. Proactive protocols are more suited for relatively stable network with fair amount of data traffic, since they perform periodic LSU flooding and provide good reliability and QoS guarantee. In these situations, one can either decide to use pure LS or OLSR (denoted as *Proactive* in the figure) under low mobility, or the more scalable HSLs (denoted as *Proactive'*) that trades off route quality for less bandwidth. The generalized hybrid protocol requires only 13 additional rules to implement, using existing declarative MANET protocols presented in Section III as building blocks.

Similar to our previous example, the threshold for adaptation (i.e. “low” and “high” values) requires additional tuning and experimentation in order to determine the appropriate values. The declarative framework and concise specifications make it easy to write other (more intelligent) switching rules in this scenario. Given a metric and threshold values, at each decision point in the tree, the adaptation rules are written in a similar manner as rules *s1-s3*.

Other policy examples are certainly possible as well, according to different network scenarios and application demands. In fact, this decision tree can be further classified, up to 8 different possibilities based on the three metrics. For instance, the choice of DTN protocol can be further refined as follows. Given a low connectivity network, a probabilistic forwarding approach such as PROPHET [15] would be suitable for a relatively high traffic and predictable mobility/dynamics scenario – this is because predictable link dynamics allow one to compute a single-copy route, albeit intermittently connected, and then *store-and-forward* data along that path. On the other hand, an epidemic protocol such as PREP [28]³ is more effective when there is a higher degree of non-predictable mobility and low traffic load.

Alternative variations to the decision tree in Figure 2 are possible. For example, in a high connectivity, low traffic and high mobility scenario, a *flooding*-based multicast strategy (in which no route computation is performed and the data packets are disseminated network-wide) can be the most effective [32].

The main point here is not to establish whether one policy is superior to another, but rather that DAWN declarative framework makes such policy specifications concise and flexible, hence enabling us to rapidly explore the rich design space of

various possible policy configurations.

V. POLICY-BASED ADAPTIVE FORWARDING

In Section IV, we show how DAWN can perform *protocol switching* at each node following a policy-based assessment of network conditions. If the mobility/dynamics and traffic conditions have high spatial diversity, locally optimized decisions can result in a non-uniform spatial distribution of routing protocols that have been chosen to run on various nodes. For instance if a network has high link dynamics in one half of the network and no dynamics in the other half, rules in Section IV can switch the nodes in the stable half to run a proactive link-state protocol while nodes in the unstable, intermittently connected portion can be instructed to run some epidemic routing scheme.

Under such circumstances, if a packet needs to be sent across the two aforementioned portions of the network, there is a necessity of traversing the proactive/epidemic protocol boundary at one or more intermediate nodes in the network. Similarly, if a portion of the network is running reactive routing, the data and control plane mechanisms⁴ of various routing protocols in use need to be coordinated at dynamically chosen “gateway” nodes located at the protocol boundaries.

In this section, we propose *inter-protocol forwarding* (IPF) techniques at the control and data plane to enable packets traversing seamlessly across different protocols, by bridging various categories of MANET routing (i.e. proactive, reactive and DTN) in DAWN. In all our examples, we define *gateway nodes* to be adjacent nodes running different protocols. Unlike prior work [7], DAWN does not require pre-selected gateway nodes. Rather, the identification of such nodes and policies for forwarding packets from one protocol type to another are expressed as declarative networking rules. Prior work on declarative networking has focused primarily on programmability/customizability in the control plane. IPF is the first usage of declarative networking in the data plane.

Our IPF mechanism assumes some degree of spatial locality in metrics such as network connectivity, mobility, and traffic patterns among nodes in the networks. This means that nodes tend to exhibit behavior similar to that of its neighbors, e.g. highly mobile nodes tend to be clustered together, while nodes that are relatively static are also surrounded by nodes of a similar static nature. This assumption ensures that any routing policies (such as the generalized hybrid protocol in Section IV-B) designed with these metrics will result in clusters of nearby nodes running the same protocol. As we demonstrate later in this section, while this assumption is not strictly required for the correct operation of our IPF techniques, making this assumption typically results in better performance as nodes can utilize a common forwarding mechanism within a cluster, and transition to other mechanisms across clusters.

In the rest of the section, we first demonstrate how IPF rules in general are implemented in DAWN, and then describe the IPF mechanisms in two steps. First, we consider only forwarding of packets in a network consisting of clusters of proactive and reactive nodes (Section V-B), followed by a combination

³PREP is an extension of the basic epidemic protocol, in which transmissions of messages are prioritized based on the residual lifetimes of individual messages. This can be done elegantly in *NDlog* by sorting the *message* table based on a user-defined ranking function, and then have forwarding rules that take the *message* table as input, and transmit based on the sorted order.

⁴These include route discovery/response, source routing, LS dissemination, next-hop routing, store-and-forward routing etc.

of proactive/reactive nodes together with disconnected nodes using store-and-forward mechanism (Section V-C).

A. Forwarding Rules

DAWN uses *NDlog* rules to easily capture various types of forwarding in the data-plane. To illustrate, the following *NDlog* rule recursively forwards a message packet along computed forwardingTable (Section III-A):

```
f1 eMessage(@N,I,S,D,P) :- forwardingTable(@R,D,N),
    R!=D, eMessage(@R,I,S,D,P).
```

Rule *f1* forwards a message packet with identifier *I* and payload *P* along the best path from source *S* to destination *D* via the next hop *N* node as computed in the forwardingTable. This payload is recursively routed by rule *f1* to the destination. The source routing rule is very similar, where the entire path is included in the initial *eMessage* and used recursively to forward the packet until the destination is reached. In Section III-C, we show an example based on epidemic-style store-and-forward mechanism of disseminating message packets.

B. Proactive and Reactive

We first consider IPF in a network consisting of nodes using either proactive (denoted as *P*) or reactive (*R*) routing. We consider both routing from a reactive node to a proactive node in the network (*R to P*) and vice versa.

R to P: When routing from a reactive to a proactive part of the network, the IPF mechanism closely resembles that of DSR protocol with caching [13]. When a route request (see *dsr1-4* in Section III) reaches a proactive node that knows the route to the final destination (typically, the destination node is also within its proactive network), a route reply can be immediately sent back to the requesting node.

```
rp1 eRouteReply(@N,S,D,P,P,C) :-
    useProactive(@N), eRouteReq(@N,S,D,P1,C1),
    bestPath(@N,D,P2,C2),
    P=f_concatPath(P1,P2), C=C1+C2.
```

To illustrate this policy, in rule *rp1*, when proactive node *N* (denoted by *useProactive*) receives *eRouteReq* originated from source *S* and finds that the best route from *N* to destination *D* has already been computed in proactive routing table *bestPath*, a *eRouteReply* will be generated and returned, in which the path from *S* to *D* is the concatenation of the path from *S* to *N* and the best path from *N* to *D*.

The salient point here is that the execution of the IPF rules determines the identity of the best gateway node *N* that the *S* to *D* path needs to pass through. If the reactive protocol is based on source-routing, *S* can send the data packet along to intermediate node *N* following the computed source route, and *N* then uses its proactive routing table to send the data packet towards *D*. On the other hand, if the reactive protocol sets up routing tables like AODV, after *S* gets a route reply from *N*, it sends the packet toward *N* using the reactive routing table, and then *N* uses proactive routing as mentioned before.

Note that DAWN nodes rely on tuples communicated among them to carry the protocol type information implicitly via the tuple name. Each node includes a set of customized forwarding rules for handling messages based on their protocol type. For

instance, in the above example, *N* receives an *eRouteReq* tuple from its neighbor. Rule *rp1* is then used by *N* to correctly respond to a request from a reactive protocol.

P to R: To enable *P* node initiating communications with unknown *R* node, one approach we explore is to equip *P* node with reactive route discovery rules (*dsr1-4* in Section III). When a proactive node (that initiates the packet forwarding) does not know the route to the destination node, a route request is issued. To reduce route discovery overhead incurred by network-wide flooding, one technique is *bordercasting* [11] route requests to border nodes within the same *P* network along already computed proactive routes. The border nodes can be defined using additional *NDlog* rules, e.g., choosing the nodes with the largest hopcount in the routing table.

Each of the border nodes then floods the route request (within the reactive networks they are attached to) and returns the best path information back to *S*. *S* then sends the data packet to the border node that minimizes the total routing cost. Note that without any coordination between border nodes, this may result in multiple route request floodings within the reactive network. This can be mitigated by additional *NDlog* rules that suppress flooding of an already-encountered packet.

Unlike traditional hybrid protocols [11], [29], DAWN does not require constraining nodes into specific routing zones, and hence provides greater flexibility on protocol adaptation.

C. Proactive / Reactive and DTN

We next consider a network consisting of proactive, reactive, and DTN nodes. We consider the scenario where DTN nodes utilize epidemic routing as described in Section III-C. Unlike proactive/reactive protocols, epidemic routing does not require the use of any routing tables, but performs opportunistic data forwarding. To illustrate how packets can be forwarded in such a network, we consider the IPF mechanisms required for routing between nodes running proactive/reactive protocol (*PR*) and epidemic routing (*E*).

E to PR: We first consider the scenario where a data packet traverses from *E* network to *PR* network. Starting from epidemic node, the packet is first forwarded opportunistically until a *PR* node is reached, at which point the packet is either directly sent to the destination (if the route is known), or trigger a route discovery process. Note that the destination node may receive multiple copies of the same packet, since the packet may be received by multiple *PR* nodes at the boundaries of the two networks.

PR to E: We next consider the reverse scenario where a packet is routed from a node (source *S*) in the *PR* network to *E* node (destination *D*). In this process, *S* first queries its routing table for *D*. If the entry does not exist, a route discovery is initiated from *S*. Upon reaching a border node⁵ in the *E* network, a route reply is sent back to *S* requesting for the actual data packet to be sent to the *E* node, at which point, epidemic forwarding is used starting within the *E* network until the packet arrives at *D*. This mechanism may also result in the destination receiving

⁵The definition of a border node can itself be customized as policy rules. In our implementation, it is the first *E* node that receives the route request enroute to destination.

multiple copies of the same packet; however the summary vector mechanism that is an integral part of epidemic routing ensures that a packet is not transmitted on any intermittently connected link more than once.

```
pre1 eRouteReply (@Z, S, D, P, P, C) :-
    eRouteReq (@N, S, D, P1, C), useEpidemic (@N),
    Z=f_last (P1), P=f_concatPath (P1, *).
```

To illustrate this forwarding process in the form of a *NDlog* rule, we consider the scenario where an *E* node at the boundaries of a *PR* and *E* network first receives a route request to destination *D*. Given that the packet is now within an *E* network, the actual data packet is required for further forwarding within the *E* network. In this case, the border node utilizes the above rule *pre1* to send a route reply back to *S*, appending a special *** symbol at the end of the reply path *P*, indicating that the packet be forwarded via route *P* to the border node for further epidemic forwarding. Note that the requesting *S* node may receive multiple such replies, and to avoid duplication, only one of the routes (typically the shortest) is chosen to forward the packet on to the *E* network.

To deal with possible message losses when *PR* and *E* networks are disconnected, one may utilize additional mechanisms, e.g. for the *S* node to buffer the original packet and resend it upon a timeout. Intermediate *PR* nodes may also perform similar buffering and resending of route requests. When a packet traverses from a *PR* to a *E* network, the packet may be sent back to nodes in the *PR* network, and in the worst case, flooded back to the entire *PR* network. To prevent this from happening, *PR* nodes can be augmented with *NDlog* rules that maintain temporary state to remember packets seen in the recent past. This consumes additional memory but reduces communication overhead.

The performance of the above mentioned hybrid protocols are somewhat dependent on the actual network topology. While it is easy to come up with MANET topologies that can result in high network overhead (e.g., ones where there are a large number of dynamically appearing border nodes between *E* and *PR* networks, and the overhead is high without implementing a coordination mechanism between the border nodes), there are useful and practical locally stable topologies where good performance can be achieved by IPF, as demonstrated in our evaluation section.

VI. EVALUATION

In this section, we perform an experimental evaluation to validate the benefits of policy-based adaptive MANETs via declarative networking. We have developed a prototype for DAWN based on the RapidNet declarative networking engine [4], [22], an open-source platform that we have developed and released. The RapidNet system is implemented as an add-on to ns-3 [1], an emerging discrete-event network simulator aimed to replace ns-2. ns-3 emulates all layers of the network stack and supports (configurable) packet loss, packet queuing, and network topology models. ns-3 supports a simulation mode enabling the controlled examination of DAWN's performance under various network topologies and conditions. Moreover, RapidNet is able to run in implementation mode on ORBIT testbed. Note that the simulation and implementation

mode utilize an identical code base, declarative rules, and rule execution engine. The only distinction is the use of network simulation in ns-3 vs *raw* sockets on ORBIT.

Our evaluation is divided into three parts. First, we evaluate a specific instance of policy-based adaptive routing based on a hybrid link-state protocol (Section IV-A) in simulation mode. Second, we evaluate in simulation the *generalized hybrid protocol* (Section IV-B) and explore the use of inter-protocol forwarding techniques to forward packets across nodes running proactive, reactive, and epidemic protocols. Finally, we deploy and evaluate the generalized hybrid protocol on the ORBIT testbed to validate our simulation results.

A. Hybrid Link-State Routing

Our first evaluation study involves *Hybrid-LS*, a policy-driven link-state protocol described in Section IV-A. *Hybrid-LS* utilizes traditional *LS* routing when the network is stable, and *HSLs* when unstable. We configure all nodes to execute *Hybrid-LS* protocol, and fix network connectivity and traffic load, while varying only node mobility. Focusing our initial evaluation on a subset of the generalized hybrid protocol enables us to study in detail effects of policy-based adaptation and protocol switching between *LS* and *HSLs*.

We conduct our experiment in a simulated wireless network consisting of 30 nodes randomly located in a $750m \times 750m$ arena. In all our experiments, nodes execute a RapidNet engine, but communicate via ns-3's simulation mode. Our simulated wireless network is based on ns-3's 802.11b PHY+MAC model and the wireless transmission range is set to 100m. By default, simulations do not use RTS/CTS among nodes, but permit up to 3 retries at the MAC layer to transmit each packet. Network simulations enable us to evaluate the performance of policy-based adaptive protocols in a controlled environment where we can vary network topology, traffic, and mobility. This complements our ORBIT testbed results in Section VI-C and reference [16].

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 2 seconds, corresponding to the flooding period in *LS* and the nominal period T_e in *HSLs*. We observe higher packet losses when the interval is increased. This is because at larger intervals, each LSU packet is also larger (since link updates are batched during the time interval for sending). Transmitting these larger packets results in increased collisions in the network.

In addition to periodic batched updates, there is network-wide refresh of LSUs 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.

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 4 m/s (on average), and *fast* stage, in which nodes move at a speed from 12 m/s (on average). Figure 3 shows the mobility setup in terms of link events per second in a total time horizon of 500 seconds (the first not-shown 10 seconds are network initialization stage). 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 2nd, 4th and 6th intervals), link events occur at a frequency of 8.8 events/second on average in the entire network. At fast speed (final interval), link events increase to 31.3 events/second. Link events during stationary stage are caused by the instability of wireless links.

Average link AA: The *link average availability* (AA) metric (first introduced in Section IV-A) reflects the stability of each link, expressed as the fraction of time a link has been up in the recent past. Each node can then compute the average AA by averaging across all neighbor link AAs. The higher the average AA, the more stable the 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 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 link AA approaches 1, and at periods of moderate speed, the average link AA can drop to as low as 0.64, only to recover when the network is stable again. At fast speed, we observe that link AA drops to as low as 0.34.

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 5-6 compare *LS* and *HSLS* based on the per-node bandwidth utilization (KBps), and one measure of route quality – *stretch*. The respective averages are summarized in Table II in the first and second columns, with another measure of route quality – *validity*.

Figure 5 shows per-node bandwidth utilization (KBps) obtained for *LS* and *HSLS* during the corresponding experimental run. Not surprisingly, *LS* incurs higher communication overhead compared to *HSLS*, averaging at 13.8KBps compared to 9.3KBps 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 IV-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 a route stretch of 1.13, compared to *HSLS* which achieves as high as 1.33. In the *stationary* stage, both protocols eventually converge to a stretch of 1, though *LS* is able to recover much quickly.

We make a similar observation based on the route validity metric, as shown in Table II, where *LS* is able to achieve higher percentage of valid routes at moderate speed, and close 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 more desirable given that it achieves routes of equivalent quality with lower communication overhead and significantly higher packet delivery rates.

Mobility	Performance	LS	HSLS	Hybrid-LS
High	BW (KBps)	35.35	17.99	21.13
	Stretch	1.16	1.14	1.09
	Validity	55.8%	50.3%	52.7%
Moderate	BW (KBps)	18.01	9.58	18.01
	Stretch	1.02	1.19	1.02
	Validity	79.8%	68.2%	79.8%

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

Benefits of Hybrid-LS: To validate that the specified policies of *Hybrid-LS* can indeed adapt between *LS* and *HSLS* based on computed average link AA, we perform a similar evaluation study by measuring the performance characteristics of *Hybrid-LS*. In this protocol, the specified policy sets the *THRES* parameter described in Section IV-A to 0.64. *THRES* can be tuned either experimentally or by analysis, and adaptive tuning of this value is an interesting avenue for future work.

Figure 7 shows the effects of this policy, by measuring the percentage of nodes using *LS* 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 average AA is above the threshold (shown in Figure 4), all nodes adapt their protocol to utilize *LS*. Conversely, after time 425 seconds, as the network becomes unstable, nodes start adapting to using *HSLS*. Eventually, all nodes adapt to using the *HSLS* protocol. At the end of the experiment, as the network stabilizes, nodes begin to re-adapt to using *LS*. Overall, we note that *Hybrid-LS* is able to effectively adapt based on network conditions.

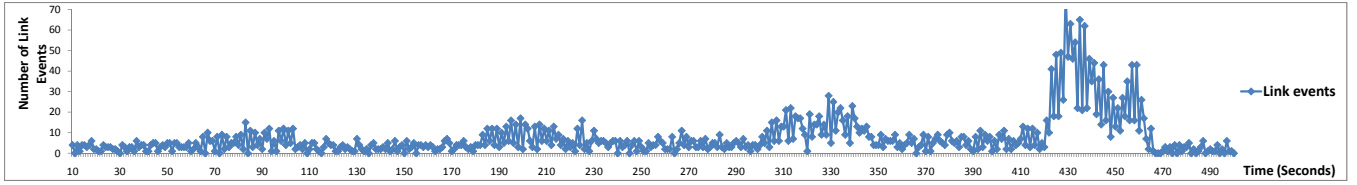


Fig. 3. Number of link events per second, with alternating 60 seconds periods of stability and mobility.

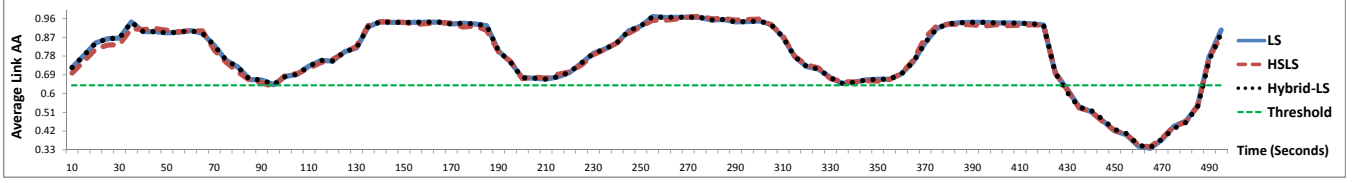


Fig. 4. Average link AA for *LS*, *HSLS* and *Hybrid-LS*. The green line indicates the switching threshold $THRES$ by the adaptation policy.

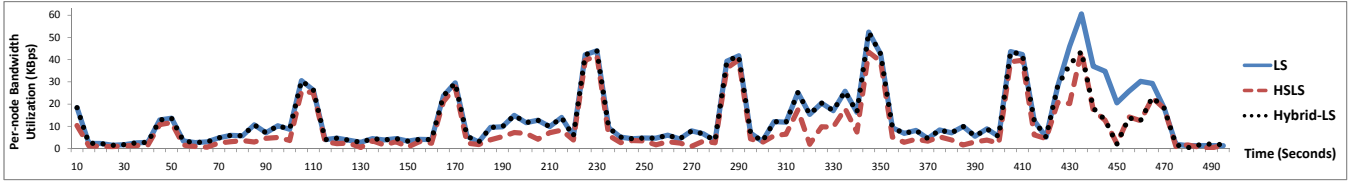


Fig. 5. Per-node communication overhead (KBps) for *LS*, *HSLS* and *Hybrid-LS*.

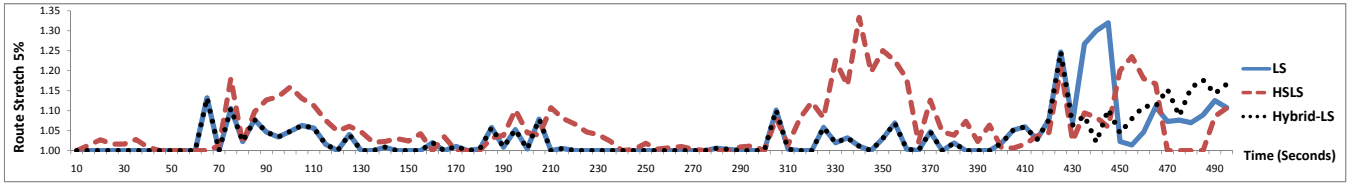


Fig. 6. Route stretch for top 5% longest routes for *LS*, *HSLS* and *Hybrid-LS*.

The third column of Table II 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*. 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 fraction of the bandwidth that *LS* requires.

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. Similarly, policy-based protocol adaptation between other categories of MANETs protocols under various metrics like network connectivity and traffic can be implemented and evaluated. We expect it to demonstrate same characteristics with *Hybrid-LS*, i.e. adopting to the most suitable protocol under specified policies and achieving the best performance of pure protocols.

B. Generalized Hybrid Protocol

Our next set of experiments focuses on the generalized hybrid protocol (*Hybrid*) presented in Section IV-B, where clusters of nodes adapt to use specific protocols based on

the policy decisions. We further evaluate the inter-protocol forwarding (IPF) techniques presented in Section V, by sending packets across nodes running different protocols. Our evaluation here consists of two parts: first a network consisting of proactive and reactive nodes, and second, a similar network that additionally includes DTN nodes using epidemic routing. Both parts are carried out in RapidNet's simulation mode.

Proactive and Reactive: Our first *Hybrid* experiment consists of a network of 28 wireless nodes randomly distributed in a 900m $\times$ 450m arena. The network consists of the following: (1) 7 low-traffic nodes at each side of the arena moving with a speed of 1-2 m/s based on the random waypoint model; and (2) 14 high-traffic stationary nodes at the center of the arena. By executing the generalized hybrid protocol, based on the policies presented in Figure 2, nodes at the arena edge use a reactive protocol (in this case, *DSR*), whereas nodes in the center adapt to use the *LS* protocol.

Our experiment is carried out over a period of 600 seconds, in which the nodes executing reactive protocol initiate communication with the proactive nodes. In total, the reactive nodes initiate 180 different route requests to discover routes to the proactive nodes in the arena center. As a basis for comparison, we compare *Hybrid* with pure *DSR* and *LS*.

Figure 8(a) shows the CDF of route request latency (in seconds) of *Hybrid* and *DSR*. Latency is computed as the time duration from the moment a source node issues a route request until a route reply is received by the same node. Overall, we observe that *Hybrid* results in lower request latency, since most

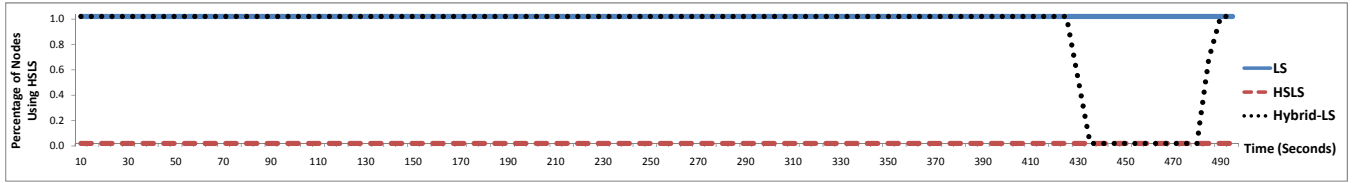


Fig. 7. Percentage of nodes using LS for *LS*, *HSLS* and *Hybrid-LS*.

route requests to proactive nodes can be satisfied by the closest proactive node to the initial requesting node. For instance, *Hybrid* has a median latency of 0.59 seconds compared to 1.31 seconds for *DSR*. Here, we have deliberately omitted *LS* in the CDF, since the protocol proactively builds routing tables and hence does not issue explicit route discovery requests.

In terms of communication overhead, *Hybrid* incurs a lower per-node communication overhead (aggregated for the duration of the experiment) of 58.99KB compared to 66.82KB of *DSR*. The lower overhead is due to the fact that *Hybrid* avoids network-wide flooding of requests whenever replies can be generated by intermediate proactive nodes. On the other hand, *LS* incurs the highest aggregate bandwidth of 73.66KB, due to the aggressive flooding of LSUs whenever links are updated. Overall, we observe that *Hybrid* is able to achieve lower route request latency while incurring lower communication overhead among the three protocols.

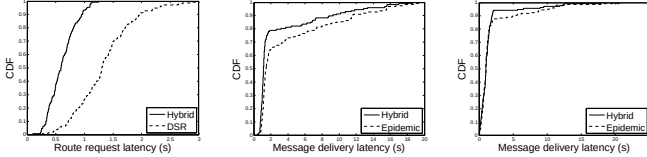


Fig. 8. (a) CDF of route request latency for *Hybrid* and *DSR*. (b) CDF of message delivery latency for *Hybrid* and *Epidemic*. (c) CDF of message delivery latency for *Hybrid* and *Epidemic* on ORBIT nodes.

Proactive / Reactive and Epidemic: Our second experiment is on a network of 30 wireless nodes in a 950m $\times$ 450m arena. Unlike the earlier setup, our experimental setup here includes additional highly disconnected DTN nodes that are using epidemic routing to transmit data packets. Specifically, the network consists of: (1) 5 disconnected nodes (*E*) at each side of the arena move with a speed of 8-16m/s following the random waypoint mobility model, and (2) 20 nodes (*PR*) at the arena center, where they execute either *LS* or *DSR*, depending on each node's traffic volume. The specific protocol is similarly chosen based on policies of the generalized hybrid protocol (denoted as *Hybrid*) shown in Figure 2.

After network initialization, data messages are transmitted from *E* nodes at one end of the arena to *E* nodes at the other end, traversing through the *PR* nodes in the center of the arena. Our experimental duration is 3300 seconds, and a total of 150 messages are transmitted, one at a time with an interval of 22 seconds. We compare *Hybrid* with a pure epidemic protocol (*Epidemic*) where all nodes utilize store-and-forward to transmit messages. In this setup, since messages from *E* nodes need to traverse through the intermediate *PR* nodes, IPF techniques (*E* to *PR* and *PR* to *E*) described in Section V-C are directly applicable. As an optimization, whenever a *PR* node discovers a border node via a route discovery, the border node is cached for a period of time and reused for subsequent

messages until it ceases to be a *E* node.

Table III compares *Hybrid* with pure *Epidemic* and *DSR*. We make the following observations. First, *Hybrid* achieves the lowest per-node communication overhead (aggregated for the duration of the experiment), since it avoids network-wide flooding of packets (*Epidemic*) or route requests (*DSR*). Second, both *Hybrid* and *Epidemic* achieves high delivery ratios, defined as the percentage of messages that are successfully delivered. *Hybrid* achieves fewer losses while utilizing only 30% of the bandwidth used by *Epidemic*. On the other hand, not unexpected, *DSR* performs poorly with a message delivery ratio of merely 11.3% in a highly disconnected environment, where route requests are often dropped by nodes that are disconnected from other nodes.

Figure 8(b) shows the CDF of packet delivery latency of all messages sent by *Hybrid* and *Epidemic*. We observe that *Hybrid* improves upon the packet delivery latency compared to *Epidemic*, particularly for packets that require long delivery time. For instance, considering only messages in the top 30th percentile of delivery time, the average for *Hybrid* is 7.0 seconds compared to 10.5 seconds for *Epidemic*. We attribute this reduction to reduced packet losses and fewer transmission retries in *Hybrid*, as compared to *Epidemic*.

Performance	Hybrid	Epidemic	DSR
BW (KB)	154.58	508.80	178.53
Delivery ratio	85.3%	82.7%	11.3%

TABLE III
PERFORMANCE COMPARISONS (BANDWIDTH UTILIZATION AND DELIVERY RATIO FOR *Hybrid*, *Epidemic* AND *DSR*).

C. ORBIT Testbed Deployment

RapidNet simulation environment enables us to study the policy-based adaptive MANETS within a controlled environment. To study protocols in an actual wireless environment, we run RapidNet in implementation mode and evaluate the same declarative rules on the ORBIT testbed [2]. ORBIT evaluation enables us to study various performance characteristics (such as actual per-node CPU overhead and memory footprint, and actual wireless interference) not observable in simulations.

In reference [16], we have presented ORBIT evaluation results for two policy-based adaptive MANET routing protocols, namely a hybrid link-state routing and a hybrid proactive-epidemic protocol. Our results are consistent with that obtained in Section VI-A. Due to space constraints, we refer the interested reader to the paper for detailed experimental analysis, and instead focus our evaluation on validating our simulation results in Section VI-B.

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). By default, we configure all nodes in 802.11a ad-hoc mode with RTS/CTS off and a retry number of 3. We have selected 802.11a as it is less susceptible to interference compared to 802.11b. We utilize 30 testbed nodes within a $5m \times 6m$ grid area for our experiments. Each ORBIT machine executes an instance of a RapidNet process. One limitation of ORBIT is that given that the maximum distance between any two nodes in our experiment is about 7.8 meters, all nodes can hear the transmission signals from all other nodes. To mitigate this issue, we reduce the transmission power to $1dBm$.

Given that nodes on the ORBIT testbed are static, we emulate mobility by first running an equivalent experiment in simulation mode, and recording all neighborhood updates from the simulation traces. The neighborhood information is used to create the `link` table at each node. In each experimental run, we add and delete tuples from each node's `link` table based on mobility traces obtained from simulation runs. Since ns-3 emulation utilizes raw sockets, `iptables` are not applicable for filtering packets at the MAC layer based on each node's current set of neighbors. Instead, application-level filtering is done by each RapidNet node by filtering incoming tuples to accept only the ones from nodes that are currently in its neighbor set determined by the current stored `link` table.

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 increased likelihood of transmission collision (and hence dropped frames) since each node's neighbors may not be the ones that are physically closest on the grid. To reduce likelihood of collisions, we add a random jitter of 0 to 200 milliseconds for each transmitted packet.

Given the above ORBIT settings, we carried out a similar 30 node experiment as described in Section VI-B, consisting of 10 *E* and 20 *PR* nodes. The actual network topology and changes to neighborhood were first captured as traces in simulation mode, and replayed on ORBIT, while all 30 nodes are running the *Hybrid* protocol.

Table IV summarizes the per-node aggregate communication overhead and message delivery ratio for *Hybrid* and *Epidemic*. In addition, Figure 8(c) shows the corresponding CDF of the message delivery latency based on a breakdown by messages. We excluded DSR from our ORBIT experiment due to its low message delivery ratio. Our ORBIT results are consistent with our previous simulation results: *Hybrid* is able to achieve high delivery ratio with lower latency while consuming only a fraction of the bandwidth required by *Epidemic*.

Figure 9 shows a representative snapshot of typical CPU usage averaged over all ORBIT nodes during a deployment period from time 500 to 800 seconds for this particular experiment. The periodic spike in CPU utilization corresponds to the periodic transmission of messages in the network. We observe that RapidNet requires low computation overhead: 3.7% for *Epidemic*, and 2.4% for *Hybrid*, at maximum. Moreover, the memory footprint is only 12MB, demonstrating the efficiency of DAWN platform on ORBIT. Overall, our results demonstrate that the additional overhead incurred in executing a declarative networking engine on ORBIT is low.

Performance	Hybrid	Epidemic
BW (KB)	139.49	514.55
Delivery ratio	92.7%	97.3%

TABLE IV
PERFORMANCE COMPARISONS (BANDWIDTH UTILIZATION AND MESSAGE DELIVERY RATIO FOR *Hybrid* AND *Epidemic* ON ORBIT NODES).

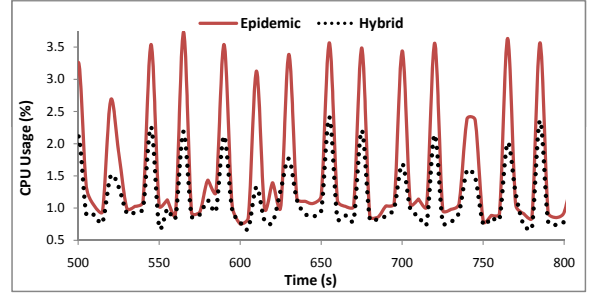


Fig. 9. Average CPU usage on ORBIT nodes running *Hybrid* and *Epidemic*

VII. RELATED WORK

Of particular relevance to our work is the literature on hybrid adaptive MANET routing protocols include Zone Routing Protocol (ZRP) [11] and SHARP [29]. ZRP hybridizes proactive and reactive protocols with routing zones, and SHARP enhances ZRP by adding adaptivity of zone radius based on traffic and application requirements. In addition, a threshold-based hybrid protocol switching between proactive and reactive routing according to network mobility is proposed in [36]. Reference [24] attempts to combine reactive AODV and DTN-style routing in hybrid scheme to achieve the advantage of both in changing network scenarios. Anxiety-Prone Link-State (APLS) protocol [26] adapts between single path link state routing in well connected parts of the network, and multi-path epidemic style routing in disrupted parts of the network.

Inter-domain MANETs [7] enables end-to-end communications over heterogeneous MANETs governed by distinct administrative domains. Their approach differs from DAWN's IPF mechanisms in Section V as follows. First there is no policy-based protocol adaptation within or across different routing domains. Second, dedicated gateways nodes are required to serve as bridges to connect heterogeneous MANETs networks. In contrast, DAWN's IPF mechanisms require no explicit gateways to be pre-specified, and forwarding policies across nodes running different protocols are customized via declarative rules.

Previously, declarative networking has been studied primarily in wired environments, such as IP routing [6] and overlay network construction [19]. Recent work [8] 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.

This paper builds upon our prior work on applying declarative techniques to enable policy-based adaptive MANET routing [18], [16]. Extensions that we have made in the paper include: (1) the generalized hybrid protocol presented

in Section IV, (2) policy-based inter-protocol forwarding in Section V, and (3) performance evaluation in ns-3 simulations and on the ORBIT testbed to highlight the above extensions. Moreover, the implementation described in this paper is based on the open-source RapidNet declarative networking engine [4], [22], which is an improved implementation over the earlier P2 system [3] presented in our prior work.

VIII. CONCLUSION

In this paper, we present DAWN, a declarative platform that creates highly adaptive policy-based MANETs. We demonstrate that a variety of MANET routing protocols can be specified concisely as declarative networks. Moreover, policy-based adaptation can be expressed in the same declarative language to create hybrid adaptive protocols based on various network and traffic conditions. We further propose inter-protocol forwarding techniques that ensure packets are able to seamlessly traverse across clusters of nodes running different protocols selected based on their respective policies. We have developed a full-fledged implementation of DAWN using our open-source RapidNet declarative networking system and experimentally validate policy-based adaptive MANETs in dynamic settings using a combination of ns-3 simulations and deployment on the ORBIT testbed. Our evaluation demonstrates that DAWN is a practical approach for deploying MANET protocols, achieving concise specifications, policy-based adaptation, and inter-protocol forwarding at low performance overheads.

IX. ACKNOWLEDGMENTS

This work is supported in part by NSF grants CCF-0820208, CNS-0831376, CNS-0845552, and CNS-1040672.

REFERENCES

- [1] Network Simulator 3. <http://www.nsnam.org/>.
- [2] ORBIT Wireless Network Testbed. <http://www.orbit-lab.org/>.
- [3] P2: Declarative Networking. <http://p2.cs.berkeley.edu/>.
- [4] RapidNet. <http://netdb.cis.upenn.edu/rapidnet/>.
- [5] P. Alvaro, T. Condie, N. Conway, K. Elmeleegy, J. M. Hellerstein, and R. Sears. Boom analytics: exploring data-centric, declarative programming for the cloud. In *EuroSys*, 2010.
- [6] Boon Thau Loo and Joseph M. Hellerstein and Ion Stoica and Raghu Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *SIGCOMM*, 2005.
- [7] C.-K. Chau, J. Crowcroft, K.-W. Lee, and S. H. Wong. Inter-domain routing for mobile ad hoc networks. In *MobiArch*, 2008.
- [8] 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*, 2007.
- [9] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). In *RFC 3626 (Experimental)*, October 2003.
- [10] H. Gill, T. Saeed, Q. Fei, Z. Zhang, and B. T. Loo. An Open-source and Declarative Approach Towards Teaching Large-scale Networked Systems Programming. In *SIGCOMM Education Workshop*, 2011.
- [11] Z. J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *IEEE Int. Conf. on Universal Personal Comm.*, 1997.
- [12] 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.
- [13] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353, 1996.
- [14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM TOCS*, 18(3), 2000.
- [15] A. Lindgren, A. Doria, and O. Scheln. Probabilistic routing in intermittently connected networks. In *MobiHoc*, 2003.
- [16] C. Liu, R. Correa, X. Li, P. Basu, B. Loo, and Y. Mao. Declarative policy-based adaptive MANET routing. In *ICNP*, 2009.
- [17] C. Liu, X. Li, S. C. Muthukumar, H. Gill, T. Saeed, B. T. Loo, and P. Basu. A Policy-based Constraint-solving Platform Towards Extensible Wireless Channel Selection and Routing. In *PRESTO Workshop*, 2010.
- [18] C. Liu, Y. Mao, M. Oprea, P. Basu, and B. T. Loo. A declarative perspective on adaptive manet routing. In *PRESTO workshop*, 2008.
- [19] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing Declarative Overlays. In *SOSP*, 2005.
- [20] Y. Mao, B. T. Loo, Z. Ives, and J. M. Smith. MOSAIC: Unified Platform for Dynamic Overlay Selection and Composition. In *ACM CoNext*, 2008.
- [21] S. C. Muthukumar, X. Li, C. Liu, J. B. Kopena, M. Oprea, R. Correa, B. T. Loo, and P. Basu. RapidMesh: declarative toolkit for rapid experimentation of wireless mesh networks. In *WINTech*, 2009.
- [22] S. C. Muthukumar, X. Li, C. Liu, J. B. Kopena, M. Oprea, and B. T. Loo. Declarative toolkit for rapid network protocol simulation and experimentation. In *SIGCOMM (demo)*, 2009.
- [23] V. Nigam, L. Jia, B. T. Loo, and A. Scedrov. Maintaining Distributed Logic Programs Incrementally. In *ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP)*, 2011.
- [24] J. Ott, D. Kutscher, and C. Dwertmann. Integrating dtn and manet routing. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, 2006.
- [25] 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.
- [26] Rajesh Krishnan et. al. The spindle disruption-tolerant networking system. In *MILCOM*, 2007.
- [27] R. Ramakrishnan and J. D. Ullman. A Survey of Research on Deductive Database Systems. *Journal of Logic Programming*, 23(2), 1993.
- [28] 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.
- [29] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks. In *ACM MobiHoc*, 2003.
- [30] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *ACM MobiHoc*, 2001.
- [31] A. Singh, T. Das, P. Maniatis, P. Druschel, and T. Roscoe. BFT Protocols under Fire. In *NSDI*, 2008.
- [32] F. Tchakountio and R. Ramanathan. Tracking highly mobile endpoints. In *WOWMOM '01: Proceedings of the 4th ACM international workshop on Wireless mobile multimedia*, 2001.
- [33] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, 2000.
- [34] A. Wang, P. Basu, B. T. Loo, and O. Sokolsky. Towards declarative network verification. In *11th International Symposium on Practical Aspects of Declarative Languages (PADL)*, 2009.
- [35] A. Wang, L. Jia, C. Liu, B. T. Loo, O. Sokolsky, and P. Basu. Formally Verifiable Networking. In *8th Workshop on Hot Topics in Networks (ACM SIGCOMM HotNets-VIII)*, 2009.
- [36] J. Xie, L. G. Quesada, and Y. Jiang. A Threshold-based Hybrid Routing Protocol for MANET. In *4th International Symposium on Wireless Communication Systems*, 2007.
- [37] W. Zhou, M. Sherr, T. Tao, X. Li, B. T. Loo, and Y. Mao. Efficient Querying and Maintenance of Network Provenance at Internet-scale. In *SIGMOD*, 2010.
- [38] W. Zhou, O. Sokolsky, B. T. Loo, and I. Lee. DMaC: Distributed Monitoring and Checking. In *9th International Workshop on Runtime Verification (RV)*, 2009.

Changbin Liu received his B.S. degree from Tsinghua University in 2007 and M.S. Degree from University of Pennsylvania in 2009, both in Computer Science. Since fall 2007, he has been working as a research assistant in University of Pennsylvania, where now he is a Ph.D. candidate in the Computer & Information Science department. His current research interests include wireless networking, distributed systems and cloud computing.

Rick Correa graduated with a B.S. degree in Computer Science in 2004 from the University of Texas at El Paso. In 2008, he received his M.S. degree in Computer and Information Science from the University of Pennsylvania. He is currently a senior member of the technical staff at Lockheed Martin's Advanced Technology Laboratories in Cherry Hill, NJ.

Xiaozhou Li received his B.E. degree in Electronic Engineering from Tsinghua University in 2008 and his M.S. degree in Telecommunications and Networking from University of Pennsylvania in 2010. He is currently a Ph.D. student at Princeton University.

Prithwish Basu is a Senior Scientist at Raytheon BBN Technologies. His research interests include network science, several aspects of mobile ad hoc networking (MANET), and in general, theoretical aspects of networking. He holds a Ph.D. degree in Computer Engineering from Boston University and a B.Tech. degree in Computer Science and Engineering from IIT Delhi.

Boon Thau Loo is an Assistant Professor in the Computer and Information Science department at the University of Pennsylvania. He received his Ph.D. degree in Computer Science from the University of California at Berkeley in 2006. His research focuses on distributed data management systems, Internet-scale query processing, and applying data-centric techniques and formal methods to the design, analysis and implementation of networked systems.

Yun Mao is a senior member of technical staff - research at the Shannon Laboratory in AT&T Labs - Research. He received the Ph.D. degree in computer and information science at University of Pennsylvania in 2008. His current research interests include distributed systems, including protocol design, programming environment, performance, and resource management.