Automated MAC Protocol Generation for Dynamic Topologies

Jian Zhen and Volkan Rodoplu

Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106

Abstract—We formulate the automated MAC protocol generation problem for dynamic topologies, as encountered in wireless ad hoc networks, under multiple neighborhoods and in the presence of acknowledgments. The probability distribution over the set of local topologies encountered in the global network serves as a model for which an optimization program may be formulated that takes the per-node average throughput as its objective function. Since the state space of the model is very large, the main challenge is the generation of the optimization program via automated methods. Symbolic Monte Carlo simulation is used to generate the optimization program, which is subsequently solved via state-of-the-art nonlinear solvers. A quantitative comparison with RTS/CTS provides information on the value of side information on the probability distribution of local topologies, which RTS/CTS does not presume. The results on computational complexity show that the time to generate the program dominates over the time to solve the resulting non-linear program, and that the complete program can be solved within reasonable computational time.

Index Terms—medium access control, network protocol, design automation, dynamic topologies, multiple neighborhood, symbolic Monte Carlo

I. INTRODUCTION

The main challenge in automatically generating MAC protocols [1] and [2] is to properly define the “protocol space” which requires modeling the impact of the exchange of control packets on the nodes’ decisions.\footnote{Prior works such as [5], [6], [7], [8], [9], [10] do not have the key feature of incorporating the impact of control information exchanges into protocol design automation, the main novelty introduced in [1] and [2].} This problem was solved in [3] for a single-neighborhood MAC channel for any number of nodes under no acknowledgments by building a compact description of the global state space of the network. Even though this model could be solved on MATLAB within minutes, this approach did not generalize to multi-neighborhood MAC models, and to the generation of acknowledgments.

In a recent paper [4], this problem was solved by a new key technique, called “Symbolic Monte Carlo Simulation”, that dramatically reduces the complexity of protocol generation, while providing a smooth trade-off between accuracy and computational complexity. As opposed to the well-known numerical Monte Carlo simulation in which numerical values are collected during the simulation, symbolic expressions that build up to and approximate the objective function are collected as the program traverses the global state space of the network without explicitly generating it. The approximation of the objective function built this way is then plugged into the non-linear solver that returns the optimal probabilities for node actions that specify the complete protocol. As an example, it takes only 3 minutes on MATLAB to generate an optimal MAC protocol (that incorporates acknowledgments) for a 9-node, multi-neighborhood network, while achieving a 10% margin-of-error on the value of the optimal throughput.

The ability to generate a MAC protocol this fast and to offer a smooth trade-off between optimality and complexity as a simple part of the design is a significant advance over the hand-design of MAC protocols.

The main shortcoming of [4] was that the symbolic Monte Carlo simulation operated on the global topology of the network; that is, it was assumed that the designer knew in advance, the graph (or topology) of the network, that is, the configuration of its set of links. While this model holds true for mesh networks (whose topologies are static), it does not hold true for wireless ad hoc networks (whose topologies are dynamic).

On the other hand, the prevalent RTS/CTS protocol in wireless ad hoc networks possesses the key strength that it makes no assumptions about the global topology of the network. Its key abstractions are (1) the identicality of the nodes, and (2) the assumption that a node does not know a priori (i.e. before deployment) into what position in the network graph it will “fall”, or put another way, in what position in the network graph it will “find itself” in the course of the evolution of the network. However, the weakness of the RTS/CTS protocol is that even though its performance is well-analyzed, no claims about its “optimality” can be made, since the formulation of the optimality of such protocols, which requires that the a priori information (which is assumed to have been attained at no cost, i.e. “side information”) be carefully modeled, has not been carried out in the past.

The contribution of this paper\footnote{This work was supported in part by the National Science Foundation grant # 0917052.} is two-fold: First, it describes how to define a measure of optimality for protocols that are designed for dynamic topologies. The key notion is that optimality is always with respect to a model of the network. Our model of a dynamic network consists of a set of local...
topologies formed around each node, and the frequency with which each of these local topologies appears in a global network. We call the set of local topologies the codebook (in analogy with a codebook, e.g. for vector quantization). Second, this paper solves the optimal MAC protocol generation problem in this setting with multiple neighborhoods and acknowledgments, for dynamic network topologies.

The rest of the paper is organized as follows: in Section II, we introduce an efficient way to represent the dynamic wireless network by using a codebook. In Section III, we present our framework for automatic protocol generation for dynamic topologies, including the multi-neighborhood MAC model, and discuss symbolic Monte Carlo simulation as well as program generation and optimization in this setting. In Section IV, we display and discuss our simulation results, and in Section VI, we present our conclusions.

II. REPRESENTATION OF DYNAMIC WIRELESS NETWORK WITH A SUBGRAPH CODEBOOK

The problem setting is given as follows: $N$ identical nodes, $n_i \in V = \{n_1, n_2, \ldots, n_N\}$ with the node index $1 \leq i \leq N$, are deployed onto a (two-dimensional) deployment region. We assume that the transmission range, $R_{TX}$, and the interference range, $R_{ITX}$, are much smaller than the size of the deployment region, which allows for a multi-neighborhood MAC. Thus, the $N$ nodes form a graph, $G = (V, E)$ with vertex set $V$ and edge set $E = \{(n_i, n_j)|1 \leq i < j \leq N\}$. For simplicity of mathematical exposition, we assume that all the nodes are slot-synchronized.

We assume that each node always has data to send to all of its neighbor nodes (that is, its transmission buffers are never empty). We also assume that each node has no knowledge about any other node unless it receives control information via successful reception of a control packet from that node. We assume that the network is “dynamic”, i.e., each node $n_i \in V$ moves continuously within the deployment region, with a random speed and direction, which can change with respect to time (indexed as $t$ with $0 \leq t \leq T_O$, where $T_O$ is the “network operation time” or simulation duration). Depending on the nodes’ movement patterns, the links between neighboring nodes can break and form.

We add the subindex $t$ to $G$ and to $E$ to denote the network graph at different times; hence, $G_t = (V, E_t)$ denotes the complete network that evolves over time. (We assume that no nodes are lost or added to the network during the network operation time or simulation duration.) We define a “centered $k$-hop graph”, denoted by $M(V, E; c, k)$ as a graph with a node $c$ (called the “center” of the graph) and a set of nodes $V = \{m\}$ and a set of edges $E = \{m, m'\}$ such that hop_count$(c, m) \leq k$ and hop_count$(c, m') \leq k$.

Based on the above assumptions and definitions, we define the “$k$-hop centered subgraph” of $n_i$, denoted by $G_{i,t}^k(i) = (V_{i,t}^k, E_{i,t}^k; i)$ around node $n_i$ at time $t$, as a subgraph of $G_t = (V, E_t)$, with

$V_{i,t}^k = \{v : \text{distance}(v, n_i; t) \leq R_{ITX} : v \in V\} \subseteq V$, \hspace{1cm} (1)

$E_{i,t}^k = \{\{v, u\} : v, u \in V_{i,t}^k, v \neq u, \{v, u\} \in E_t\} \subseteq E_t$, \hspace{1cm} (2)

where “distance($v, u; t$)” is the physical distance between nodes $v$ and $u$ at time $t$. (Note that this may be roughly indicative of the hop count between $v$ and $u$, but not necessarily so.)

In Fig. 1, an example of a 1-hop centered subgraph is given; $G_{i,t}^1$ is the 1-hop centered subgraph centered around node $n_i$ at time $t$. The $k$-hop neighborhood subgraph of any node eventually converges to the whole network graph $G_t$, as $k$ increases. In this paper, we consider only the 1-hop centered subgraph, $G_{i,t}^1$; thus, we simply write it as $G_{i,t}$, and write $V_{i,t}$ as $V_{i,t}^1$ and $E_{i,t}$ as $E_{i,t}$, in the rest of the paper. Then, the “subgraph codebook” of $\{G_t\}_{t=T_0}$, denoted by $C_G$, is a hash table with elements $(C_G)_{key,w}$ such that each key of $C_G$ is a centered graph $M(V, E; c, k)$ and each value of $C_G$ is the fraction of the elements of $\{G_t^k(i)|0 \leq t \leq T_O, 1 \leq i \leq N\}$ that are isomorphic to $M(V, E; c, k)$.

In this paper, we are interested in the throughput performance of the network. First, we define node $n_i$’s “instantaneous throughput”, $\hat{F}_{i,t}$, as the number of received packets in one time slot, at node $n_i$ and at network time $t$. In our formulation, we let each data packet occupy exactly 1 time slot; thus, $\hat{F}_{i,t}$ is a Bernoulli random variable (with the two values 0 and 1). Once the same MAC protocol is deployed for all the nodes in the network, the probability distribution of $\hat{F}_{i,t}$ depends only on the network graph at time $t$ and node $n_i$’s position in the graph. Note that because the physical movement is very slow in comparison to the speed of packet transmission, we are justified in using a “static network” model. Then, we define the “individual node’s short-term throughput” at time $t$, denoted by $F_{i,t}$, as the conditional expectation of the node’s instantaneous throughput, namely as $F_{i,t} = E[\hat{F}_{i,t}|G_t]$. The “node’s average throughput”, $\bar{F}_i$, is defined as the time average of the node’s short-term throughput, namely as $\bar{F}_i = \frac{1}{T_O} \int_0^{T_O} F_{i,t} dt$. Finally, we define the “per-
node average throughput of the network, \( \bar{F} \), as the average of the throughputs of all the nodes in the network, namely as \( \bar{F} = 1/N \sum_{i=1}^{N} \bar{F}_i \). Putting these together, we have

\[
\bar{F} = \frac{1}{T_o} \int_{0}^{T_o} dt \left( \frac{1}{N} \sum_{i=1}^{N} E[\bar{F}_i(t)] \right)
\]

(3)

Now, we use the fact that \( F_{i,t} = E[\bar{F}_i(t)|G_t] \approx E[\bar{F}_i(t)|G_{i,t}] \), where \( G_t \) is the network graph at time \( t \) and \( G_{i,t} \) is “the centered subgraph around node \( n_i \) at time \( t \)”, which means that the knowledge of the local network around node \( n_i \) suffices in computing the expectation. (This is due to the locality of a MAC channel.) Thus, we have

\[
\bar{F} = \frac{1}{T_o} \int_{1}^{T_o} dt \left( \frac{1}{N} \sum_{i=1}^{N} E[\bar{F}_i(t)|G_t] \right)
\]

(4)

\[
\approx \frac{1}{T_o} \int_{1}^{T_o} dt \left( \frac{1}{N} \sum_{i=1}^{N} E[\bar{F}_i(t)|G_{i,t}] \right)
\]

(5)

We define the “subgraph counting function” \( A((C_G)_{key}, t) \) as \( A((C_G)_{key}, t) = \| \{G_{i,t}|G_{i,t} \cong (C_G)_{key}, 1 \leq i \leq N \} \| \), which counts the number of the subgraphs \( G_{i,t} \) at the time \( t \), which are isomorphic (“\( \cong \)” means “graph isomorphism”) to the codebook key \( (C_G)_{key} \). Recall that \( \bar{F}_{i,t} \) is the throughput of node \( n_i \) in the centered subgraph \( G_{i,t} \). (Note that \( \bar{F}_{i,t} \) is a random variable.) Note that \( \bar{F}_{i,t} \) and \( \bar{F}_{j,t} \) \((n_i, n_j \in V, \text{ and } 0 \leq t, t' \leq T_O) \) are identically distributed if \( G_{i,t} \cong G_{j,t'} \) \( (C_G)_{key} \). We let \( \bar{F}_{key} \) be a random variable with this identical distribution. Let \( K(C_G) \) denote the set of all the keys of \( C_G \). Then,

\[
\bar{F} = \frac{1}{T_o N} \int_{1}^{T_o} dt \sum_{(C_G)_{key} \in K(C_G)} E[\bar{F}_{key}|(C_G)_{key}] \cdot A((C_G)_{key}, t)
\]

(6)

\[
= \sum_{(C_G)_{key} \in K(C_G)} E[\bar{F}_{key}|(C_G)_{key}] \frac{1}{T_o} \int_{1}^{T_o} dt \left( \frac{A((C_G)_{key}, t)}{N} \right)
\]

(7)

We assume that the process \( A((C_G)_{key}, t) \) is an ergodic process. Then, the probability, with which a node finds itself as the center of the subgraph \( (C_G)_{key} \) is

\[
P[(C_G)_{key}] = \lim_{T_o \rightarrow \infty} \frac{1}{T_o} \int_{0}^{T_o} dt \left( \frac{A((C_G)_{key}, t)}{N} \right)
\]

(8)

For this probability to exist, we assume that \( A((C_G)_{key}, t) \) converges as \( T_O \rightarrow \infty \). It is worthwhile to note that this is the only assumption that we make about the network dynamics; besides this, the nodes’ movements are not restricted to any mobility model. The nodes may move with arbitrarily varying velocity (which includes changing direction).

\(^{3}\) Other metrics or weighted combinations thereof may be substituted while retaining this framework.

Based on the above assumption, we have

\[
\bar{F} = \sum_{(C_G)_{key} \in K(C_G)} E[\bar{F}_{key}|(C_G)_{key}] \cdot P[(C_G)_{key}]
\]

(9)

For simplicity, we write \( P[(C_G)_{key}] \) as \( \phi_{key} \in \Phi \), where \( \Phi \) is the probability distribution on \( K(C_G) \). Then,

\[
\bar{F} = \sum_{(C_G)_{key} \in K(C_G)} E[\bar{F}_{key}|(C_G)_{key}] \cdot \phi_{key}
\]

(10)

We see in (9) that the per-node average throughput is equal to the average of the throughputs of the subgraphs, weighted by their frequency of occurrence. We also note that a dynamic wireless network can be efficiently represented by the pair \( (C_G, \Phi) \), i.e., the subgraph codebook.

In order to demonstrate the subgraph codebook intuitively, we give a simple example in Fig. 2: This example uses a network with periodic dynamics; however, as can be seen in the development that leads up to (10), no periodicity is necessary for (10) to hold. From time \( t = t_0 \) to \( t = t_0 + 1 \), \((t_0 > 0)\), \( n_i \) is moving from the left to the right while breaking the links on its left and forming new links on its right. The probability distribution (computed as a relative frequency) on the subgraphs remains the same, despite the fact that the global topology oscillates between the two shown topologies.

III. FRAMEWORK OF AUTOMATED MAC PROTOCOL GENERATION FOR DYNAMIC TOPOLOGIES

Based on the above probabilistic model of a dynamic network, we solve the automated MAC protocol generation problem by using the symbolic Monte Carlo method [4]. The objective function we use in this paper is the “per-node average throughput” and is equal to the \( \Phi \)-weighted sum of the individual node’s throughput given, as described in the previous section, by

\[
\bar{F} = \sum_{(C_G)_{key} \in C_G} \phi_{key} \cdot E[\bar{F}_{key}|(C_G)_{key}]
\]

(11)

where \( \phi_{key} \in \Phi \) and \( (C_G)_{key} \in C_G \) in codebook \( (C_G, \Phi) \). Thus, this maximizes the expected value of the objective function (e.g. throughput) over this probabilistic model of the network. Note that a single protocol is generated.

The MAC model we use in this paper is similar to the one in our previous paper [4], so only a brief description shall
be given here. In the MAC model, only a minor difference exists, namely that we have disabled the acknowledgment for the received data packet, since it was found that this was never used in the optimal solution in the case of throughput maximization (see the results in [4]). Thus, each node takes any one of the five actions in every slot: (1) $n$; the node remains silent, (2) $d$; the node sends a data packet, (3) $a$; the node sends an acknowledgment, (4) $c$; the node sends a control packet that is not an acknowledgment, by which it tells (1-hop) neighbors that it will send a data packet for the following $W$ slots, where $W$ is the “control lifetime”, i.e. the number of slots for which the effect of the control packet on the receiving node lasts, (5) $ca$, which has the same meaning as $c$ but which requires that the receiver node send back an acknowledgment $a$ right after it receives $ca$. Note that $a$ is an acknowledgment only for a control packet; there are no acknowledgments for data packets. Once the node receives $a$, it takes hold of the channel if $a$ is destined to it and keeps silent if it is not destined to it. Thus, the actions that a node can take form the set of actions denoted by $\mathcal{A} = \{ n, d, c, ca, a \}$. We define the “initial set of actions”, as a subset of this action set, as $\mathcal{A}_i = \{ n, d, c \}$.

We define the state of a node as $< s, w >$, where $s \in \mathcal{S}$ is the “knowledge state”, which encodes that node’s knowledge regarding its control information exchanges (namely, this encodes the control packets that this node has sent, and the control packets that it has received in the last $W$ slots), and $w \in \{ 1, 2, ..., W \}$ is the state timer which counts down and records how many slots the node will still be in this state. We assume unicast data transmission and broadcast control transmission, i.e., only the destined receiver node decodes successfully the data packet $d$ that it receives, while each of the neighbor nodes may decode all of the control packets that it receives. Thus, the set of knowledge states is denoted by $\mathcal{S} = \{ \phi, s_c, s_{ca}, s_a, q_c, q_{ca}, q_a, l_c, l_{ca}, l_a \}$, where $\phi$ is the null state, in which no control information has been sent or received by the node during the last $W$ slots; $s_c$, $s_{ca}$, $s_a$ mean that $c$, $ca$ or $a$ (respectively) has just been sent in the last slot; $q_c$, $q_{ca}$ and $q_a$ mean that $c$, $ca$ or $a$ (respectively) has been received in the last slot and the packet was destined to this node; $l_c$, $l_{ca}$ and $l_a$ mean that $c$, $ca$ or $a$ has been received in last slot but was destined to another node (such a case is due to the broadcast nature of control transmission).

Upon arriving at one of these states, the node chooses an action $x \in \mathcal{A}$ with probability $P(x|s)$, according to the transition rule of this state, and starts the timer with $w = w(s)$ accordingly, and the timer counts down each slot until it triggers another state transition when it times out. The transition rules of this model are summarized in Table I. In this table, $\theta_2, \theta_3, \theta_{ca} (\sum \theta = 1)$ are the design variables (a.k.a. decision variables) of the optimization. Note that this table is not a complete transition table for the finite state machine (FSM) of a node; for simplicity, it describes only the transition rules of phase 2 for a 2-phase FSM (see [4]). The phase 1 transition rules (from actions to next state) are rather straightforward, and are not shown. In order to optimize the performance of the MAC protocol (recall that our chosen metric is throughput in this paper), we utilize the Symbolic Monte-Carlo method [4] to explore the global state space of the network; we collect symbolic terms, and accumulate the symbolic expressions for the two metrics of interest that help us compute the average throughput. These two metrics are $E[F]$, namely, average number of successful transmissions per cycle, and $E[T]$, the average length of the cycle, where a cycle is defined as the time between two subsequent visits to the same recurrent state. The long-term average throughput can be computed as $E[F]/E[T]$. The key aspect of the Symbolic Monte Carlo method is that the whole state space need not be explored; this would be a computationally intensive task even for a small network with multiple neighborhoods. Instead, the state space is sampled by running a symbolic Monte-Carlo simulation in which symbolic expressions are collected via only the sampled routes through the state space. In the end, an approximation of the objective function is obtained. Finally, after the objective function has been computed by Symbolic Monte Carlo, the resulting non-linear optimization program is solved by using the openopt package available for Python.

| state $s$ | action $x$ | $P(x|s)$ | timer $w(x|s)$ |
|-----------|------------|----------|----------------|
| $\phi$    | $n, d, c, ca$ | $\theta_2, \theta_3, \theta_{ca}$ | 1, 1, 1 |
| $s_c$     | $d$        | $1$      | $W$ (control lifetime) |
| $s_{ca}$  | $n$        | $1$      | $1$ |
| $s_a$     | $n$        | $1$      | $W$ |
| $q_c$     | $n$        | $1$      | $W$ |
| $q_{ca}$  | $a$        | $1$      | $W$ |
| $q_a$     | $d$        | $1$      | $W$ |
| $l_c$     | $n$        | $1$      | $W$ |
| $l_{ca}$  | $n$        | $1$      | $W$ + 1 |
| $l_a$     | $n$        | $1$      | $W$ |

TABLE I TRANSITION RULES OF A NODE UPON ARRIVING IN STATE $s$
tions were run on a Lenovo ThinkPad SL410 laptop, with no other concurrent computationally intensive processes.

The simulation structure is illustrated in Fig. 3: the dynamic network generator randomly generates the traces of the nodes for a dynamic network. The subgraph codebook generator takes in the traces of the nodes and builds the corresponding subgraph codebook for this dynamic network. Based on the codebook and the given MAC model, symbolic Monte Carlo simulation explores the design space and generates an approximate symbolic expression for the objective function. At the end, the values of the optimal design variables are generated by the optimizer.

The original version of the symbolic Monte Carlo method used in our previous paper [4] suffers from slow convergence. This is due to the inefficiency of random sampling: (1) many samples are picked multiple times; hence, repetitions lower computational efficiency, and (2) it is difficult to sample uniformly over the feasible set merely by setting good transition probabilities; this adds bias to the generated expressions. In order to overcome these drawbacks, we designed a “two-stage” symbolic Monte Carlo simulation. This approach accelerates the previous [3] symbolic Monte Carlo method by intelligently choosing the samples which are never duplicates of each other. In the first stage, we exhaustively search the design space using breadth-first search for the first \( D_{bfs} \) steps, starting from the null state \( \phi \), where \( D_{bfs} \) is chosen dynamically and implicitly: once the breadth-first search collects enough sample paths (> 1000 in this simulation) at the \( l \)-th step, \( D_{bfs} \) is set to \( l \) and we finish the first stage. In the second stage, we continue each of the sample paths built in the first stage and run the original symbolic Monte Carlo, in which each next step is chosen randomly until the path loops back to the null state. (If the path loops back to \( \phi \) within the first \( D_{bfs} \) steps, we terminate that path and keep it as one of the sample paths.)

In Fig. 4, we show the optimized design variables \( \{\theta_i\} \) under different control lifetimes, from 1 to 9. In order to make a fair comparison among different control lifetimes, we use the same nodes’ traces, generated by the dynamic network generator. We can see in this figure that for different control lifetimes the optimal design variables are different, and that the figure can be divided into three main regimes: 1) the control lifetime is 1, where no control packet (c or \( c_a \)) is sent because the benefit of successfully transmitting control packets is too small; 2) the control lifetime is 2 and 3, where all the node actions are used with different weightings; 3) the control lifetime is larger than 3, where only the control packet \( c_a \) is sent. Each of these regimes corresponds to a protocol that is structurally different from the others; hence, this provides an example as to how this framework generates structurally different protocols under the same umbrella. In the traditional hand-design of protocols, the human designer makes a (usually implicit) guess at what the underlying control lifetime is (this may be governed, for example, by the coherence time of traffic generation under dynamic traffic conditions, which is not addressed in this paper). Based on this implicit guess, a protocol is intuitively designed as to include or exclude a control packet such as \( c_a \), in-hand design. Moreover, most hand-designed protocols comprise only very elementary randomness in the choice of node actions; in contrast, above, optimal weightings according to which nodes take actions from the action set are specified and become part of the protocol description. The main reason that this can be done is that optimality that incorporates control information exchanges has been formulated a priori in our framework, whereas no such rigorous optimality framework that includes control information generation exists in the hand-design of protocols. Numerical optimizations exist for protocols whose control structure has been fixed; however, the above framework differs from these in that structurally different protocols are subsumed under the same umbrella.

In Fig. 5, we display the per-node throughput under different control lifetimes, where the control lifetime \( W \) ranges from 1 to 5. When we compare the designed MAC protocol with the widely used RTS/CTS protocol, we see that the designed MAC protocol outperforms RTS/CTS for all \( W \)'s in this range. The reason is that RTS/CTS does not utilize the distribution with which local subgraphs appear in the global network. This is both a strength and a weakness of RTS/CTS. The graph shows the extent to which this information, when available, can be utilized to improve the throughput, while relying only on local information. This type of information, as displayed in this graph, also provides a partial answer to the more general question that we raised earlier, namely, a definition of the optimality of protocols with respect to differing levels of side information. While RTS/CTS is not provably optimal for the case without the side information on the distribution of local topologies, since it is a feasible solution, it
provides a lower bound for the performance of protocols without this side information. Thus, the graph displays the performance difference between this feasible solution and the optimal solution (i.e., protocol) with the side information on the probability distribution of the local topologies. In Fig. 5, we can also see that slotted ALOHA performs the worst except for when control lifetime less than 3, in which case RTS/CTS performs worse than slotted ALOHA. This is because for the smaller control lifetime, RTS/CTS has a relatively large control overhead whereas slotted ALOHA does not.

Next, we analyze the computational complexity of the optimal protocol obtained for the same dynamic network. In Fig. 6, the computational complexities (characterized by the execution times) for the three major parts of the framework are shown. It can be seen that expression generation (i.e., state space exploration with Symbolic Monte Carlo) accounts for most of the complexity. While Python may be replaced by a more efficient language, this result points to a general result that we have seen so far, namely that the time to generate the optimization program dominates over the time to solve the program using non-linear state-of-the-art solvers.

V. ACKNOWLEDGMENTS

The authors would like to thank Forrest Brewer for the germ of the idea that led to the speed-up of Symbolic Monte Carlo method, described in paragraph 4 of Section IV.

VI. CONCLUSIONS

We have presented a novel framework for automated MAC protocol generation for dynamic topologies. The key idea in handling network dynamism is to model the network via a collection of local topologies (a.k.a., subgraphs) and their probability distribution. Empirical frequencies of these local topologies collected during simulation serve to approximate this probability distribution. An optimal protocol for a dynamic topology is defined with respect to this model. In this paper, we have solved this optimal protocol generation problem for dynamic topologies by generating a mathematical program using the symbolic Monte Carlo method. We have also quantified the value of side information on the distribution of local topologies by comparing the performance of the optimal protocol with RTS/CTS. The results on computational complexity show that the problem can be solved in a reasonable time, and that the time to generate the approximate program dominates over the time to solve the generated optimization program.

REFERENCES