TORA, Verification, Proofs and Model Checking

Shah-An Yang and John S. Baras
Department of Electrical and Computer Engineering
and the Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

EXTENDED ABSTRACT

Routing protocols for mobile adhoc networks (MANET) are one of the most essential components of such networks; the other being media access control (MAC) protocols. Unfortunately most existing routing protocols for MANETs are so complex that the investigation of their properties, weaknesses and performance cannot be done analytically (at least using conventional analytic methods); instead simulations have become the typical means of analyzing their properties and performance. This is an unfortunate state of affairs for such an important networking component. In this paper we develop a different means for analyzing the properties and performance of routing protocols for MANETs based on methods and techniques from formal specifications [3] and associated formal means for proof and validation. The aim is to develop rigorous proofs and tools for performance checking and analysis. The primary goal of this type of research is to develop a way to automatically check a specification for correctness and liveness properties. A secondary, but equally important role is the development of automatic tools for ameliorating discovered weaknesses and for designing such routing protocols so as to satisfy prespecified properties and performance.

In this paper we use rigorous mathematical proof and model checking to verify the correctness of the Temporally Oriented Routing Algorithm [1] (TORA), which is a MANET routing algorithm (protocol). The main difficulty in applying any automated formal methods is exponential state space explosion. Formal methods have been applied successfully to protocols where the number of states is clearly finite. TORA has an infinite number of states, though there is a structure to the state space that makes it simpler than the general problem.

TORA derives from a class of link reversal algorithms referred to as the Gafni-Bertsekas (GB) algorithms [2]. Any algorithm of this class exhibits certain desirable properties, but TORA falls outside this class of algorithms. For example, while all GB algorithms are path independent, TORA is not, meaning the set of final heights is not entirely determined by the initial conditions. Also, unlike the GB algorithms, the number of reversals depends on the ordering of events. While many properties from the GB algorithms are lost, TORA should still always converge in a finite period of time. Establishing this formally is one of the results of this paper.

The heights used in TORA are based on time (temporally oriented heights), and therefore they are always globally the greatest heights in the network. This can improve the performance over other link reversal algorithms of the GB class. Consider the case of ordinary partial reversal algorithms and consider a chain of nodes where the heights are ordered completely backwards with respect to the location of the destination. In the case of TORA, the local minimum at the end of the chain would define a new globally highest reference level. The nodes in the chain upstream of TORA would then have room to increase their heights without exceeding the new globally highest node. In the case of non-temporally oriented heights, this is not the case and a large number of "oscillators" are necessary before all the heights converge.

Another difference between the GB algorithms and TORA is behavior under network partition. In GB algorithms the heights (the distance metric associated with each node) grow unboundedly under partition. In practice, this causes unproductive traffic to continually propagate. TORA includes a partition detection mechanism to prevent this as well as mechanisms for reactivating a node once it has been deactivated. TORA also features some performance improvements over the original GB algorithms. While a proof of correctness and convergence properties for the GB algorithms exists, TORA may fail to converge under some very special conditions involving changes in topology and link requests.

We attempt to verify the properties of TORA using two tools from verification: model checking and proof by hand. The hand proofs establish properties of TORA in an unpartitioned network under static topology, but do not reveal its behavior under partition. For this, we resort to model checking. Since TORA has infinite states even in finite networks, we had to perform state space reduction. We use the technique of weakening of the specification by introducing supplemental nondeterministic state transitions into the finite automaton model of the algorithm in order to reduce states that could not be reduced by any obvious method. Additionally, we examine the problem of generating topologies for modeling mobile ad-hoc networks and explore other state space reduction methods, including the exploitation of network symmetries by using graph automorphisms.

As a descendant of the simpler Gafni-Bertsekas (GB) algorithms, TORA has enough in common with the GB algorithms that it is possible to prove that TORA converges in fixed, connected networks when the destination is included with a similar proof. Convergence in this case differs from

the GB algorithm because it includes the case where TORA detects a partition, which is possible even when no partition exists. However, our proofs show that nodes of common reference level form a directed acyclic graph, rooted at the node originating that reference level and effectively perform a search for the destination in the network. As a consequence, a sufficient condition for TORA to not detect a partition in an unpartitioned network is that any node originating a new reference level once the topology has become fixed will never detect a partition.

In the case where the network is under partition, the result above does not hold. It is necessary to prove that the partition detection mechanism converges. We were not able to discover the right invariants that would yield a proof of this property. We attempted to verify this property through model checking. The advantage of a model checking is that the process of verification is more automated than in that of proof by hand. It's main disadvantages are that a model checker can only verify a property under a finite number of scenarios and also such a verification yields less insight into the workings of the algorithm than a hand proof.

To verify convergence in a MANET protocol, it is necessary to consider states reachable not only under events of the protocol itself, but also changes of topology. To verify that TORA always detects a partition in partitioned networks, it is necessary to show that it holds for all possible reachable states. The reachability question is difficult to model check because there are an infinite number of possible sequences of topology changing events. Our approach here was to assume that all states are reachable and to show that the convergence property holds from any state.

As the state of TORA depends on timestamps, there are possibly an infinite number of states to be examined. Upon careful examination of the protocol, we deduce that only the ordering of the timestamps between the nodes is significant as the timestamps are only a method of establishing a new globally highest reference level. A similar method applied to most of the other state variables. Another type of reduction required weakening the specification at the state machine level. Since the δ values in TORA can be decremented, the incremental difference between δ values in the algorithm is significant, hence to completely capture the behavior, it is necessary to model them as integers, but integers would be impossible to enumerate. We weaken the specification so that δ values are still modeled by their ordering, but when a δ value is decremented, it may nondeterministically become less than the next lower δ value in the state or equal to it.

Even though the algorithm was originally implemented in SPIN [3], it became very tedious to specify the different conditions that the algorithm would experience using SPIN's restricted syntax. SPIN adopts a concurrency model that is similar to the one used in Hoare's Communicating Sequential Processes. The most crucial part of SPIN that differentiates it from a general programming language is the non-determinism. In SPIN, all decisions and control flow must be implemented as a non-deterministic choice between guarded statements. In executing the program SPIN uses the non-determinism as a decision tree for exploration. What makes this less than ideal for TORA is in choosing the correct model for the algorithm. We are interested in modeling TORA over a wide range of varying topologies and initial conditions. To implement this in SPIN, it is necessary to construct the network topologies and all the initial conditions using the non-determinism operators that are available in SPIN. This is quite unnatural, especially in the sense that the non-determinism of the construction of the network carries the overhead of being possibly interleaved with other events. There are an infinite number of network topologies and an infinite number of different combinations of initial heights. Several initial runs in SPIN yielded poor coverage of the state space and we decided that using a general programming language would be a much more programming and execution efficient way of modeling the system than through SPIN. Another reason why SPIN is not very suitable for this particular problem is that there are too many possible changes in topology that can result in different states to model. SPIN enumerates states starting from an initial condition and applies partial order reduction to eliminate redundant interleavings of concurrent processes. This complicates the reachability analysis of TORA.

To reduce the redundancy of states checked, we used graph automorphisms on the topologies. In any orbit, we always labeled the states according to their lexical ordering. This reduces the state space exactly by a factor of the number of automorphisms in the topology under test. To further improve the speed with which we could enumerate states, the model checker was written in a distributed fashion where we took advantage of the course grain parallelism inherent in the model checking problem. The result was that we verified the convergence property for all topologies up to 4 nodes. The limiting factor was the minimum amount of RAM in any computational node.

For detailed description of the results reported here we refer to [4].

Acknowledgment

This research was supported in part through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DADD 19-01-2-0011.

References

[1] V. Park and M. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", *IEEE Proceedings of INFOCOM 97* (April 1997) : 1405-1413, 1997.

[2] E. Gafni and D. Bertsekas, "Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology", *IEEE Transactions on Communications* 29 (January) : 11-18, 1981.

[3] G.J. Holzmann, "The Model Checker Spin", *IEEE Transactions on Software Engineering* (May) : 279-295, 1997.

[4] S. Yang, *TORA, Correctness, Proofs and Model Checking*, M.S. Thesis, Electrical and Computer Engineering Department, University of Maryland, December 2002; available as Technical Report from the Institute for Systems Research.