Correctness Proof for a Dynamic Adaptive Routing Algorithm for Mobile Ad-hoc Networks

Shah-An Yang and John S. Baras

Electrical and Computer Engineering Department
and the Institute for Systems Research
University of Maryland
College Park, MD 20742, USA

Abstract

Dynamic and adaptive routing algorithms are the "brains" of mobile ad-hoc networks (MANETs), in that they govern the self organization of these networks. MANETs are infrastructureless "intelligent" networks, which are becoming increasingly popular and have wide applicability. In this paper we view dynamic adaptive routing algorithms for MANETs as hybrid systems (partly based on logic and partly based on numerics). In this paper, we introduce formal models for the analysis and verification of one such routing algorithm, the Temporally Oriented Routing Algorithm (TORA) under certain assumptions. More specifically we give a rigorous mathematical proof of correctness and convergence of TORA. Other formal methods, their limitations and future research challenges are also discussed.

1. Introduction

Mobile ad-hoc networks (MANETs) are infrastructureless communication networks that are becoming popular due to the expansion of broadband wireless connectivity to millions of users. They provide instant infrastructure for many applications including communications on the move, collaborative work, gaming, disaster relief, exploration, and defense. Routing algorithms (or protocols as they are often called) are an integral part of the operations of MANETs, particularly since they are directly involved in the self organization of these networks. Self organization is the reason why MANETs are considered "intelligent" systems. Thus careful design, evaluation and verification is a critical need for MANETs. Unfortunately, despite intensive research we do not have today many systematic and analytical methods and tools for the design, evaluation and verification of routing algorithms for MANETs. The situation is even worse regarding analysis of security properties of these routing protocols.

These routing protocols are very good examples of hybrid systems, in the sense that they have some components that are logic-based and some components that are based on numeric variables and computations. Since they form the "brains" of MANETs, it is therefore of paramount importance to develop systematic methods for the design, evaluation, and verification of routing protocols for MANETs.

In this paper we apply formal methods to the verification of TORA, which is a mobile ad hoc networking routing algorithm. The main difficulty in applying any formal methods to a system like this is exponential state space explosion. More commonly, formal methods have been applied to protocols rather than algorithms 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.

The primary reason for making things formal is that in doing so, the system becomes more precise and loses any ambiguity in interpretation. Sometimes, the only precise specification of any system is actually the source code, but code contains too many details specific to the implementation language and machine architecture. In our work, we considered several types of formal models including extended finite state machines, finite automata and regular expressions. A formal model here means that the system can always be decomposed into nothing more than typographical manipulations, which is why string representations are important. The ultimate goal of formal methods and models is to develop a way to go automatically from a specification to a proof of correctness for a system.

There are two separate approaches to formal methods although some work has been done towards their unification. The first category of formal methods is state enumeration techniques. The underlying model used here is that of a finite automaton. The typical state enumeration system requires the specification of the system in some formal language, each claiming some advantages over the others, but fundamentally, they are all exhaustive simulations with extensive state space pruning techniques. The limitation in these approaches is state space explosion. In a routing algorithm like TORA, there is an infinite amount of space and an infinite number of scenarios that it can act in. At most, a model checker will be able to say that under a given finite set of scenarios, a routing algorithm like TORA behaves as expected. There is not in general a way to automatically infer that based on correctness in a finite number of scenarios that the algorithm performs correctly in an infinite number of cases. It is usually unknown whether the very next unexplored scenario will exhibit an incorrect behavior.

The second approach can be called proof methods. This approach attempts to deduce properties of the system by using theorems. There is some work that has been done towards automating this, but most work is still done by hand. Even so called "automatic theorem provers" would more aptly be called automatic theorem "checkers" as their main strength is in making sure that the human theorem prover remains honest and does not make any mistakes in his calculations. Verification of proof steps is decidable in general, but actually proving a result from axioms is not. This makes it possible to check theorems automatically, but not prove them. Using theorems and proofs is much more powerful than model checking in the sense that it is possible to establish general results for an infinite number of scenarios. The disadvantage is that very little can be automated and a large amount of the work is left to the human analyst.

1.1 *TORA*

TORA, like other distance vector algorithms, uses only local information to maintain global structure. The information is distributed across different nodes, and no individual node has complete information about the routes in the network. Each individual node acts according to a set of simple rules and through their combined behavior, routes emerge. The goal of this work is to "decompile" the lower level mechanisms of TORA into higher level mechanisms that can be verified as producing routes for the network.

TORA is based on a class of algorithms referred to as the Gafni-Bertsekas (GB) algorithms. This class of algorithms is deficient in that when the network is partitioned, the heights (the distance metric associated with each node) grow unboundedly. In practice, this will cause excessive network traffic as routing information continually propagates unproductively. TORA includes a partition detection mechanism to prevent this from happening. This includes mechanisms for reactivating a node once it has been deactivated. TORA also features some performance improvements over the original GB algorithms.

A proof of correctness and convergence properties for the GB algorithms exists. However as we shall see shortly TORA is not a GB algorithm. Indeed, there are actually cases where TORA fails to converge under some very special conditions involving changes in topology and link requests.

1.2 *Notation*

Throughout this paper, the following notation is used.

G	a finite graph with undirected links
V	the set of ordered vertices or nodes in G
E	the set of edges or links in G
N	the size of vertex set $\|V\|$
x	a node in V (each node x is assumed to have a unique integer ID and x is used interchangeably to represent
(x, y)	a link in V ((x, y) is equivalent to (y, x) as all links are assumed undirected)
$N(x)$	neighbors of x, $\{ y \in V \mid (x, y) \in E \}$
$D(x, y)$	length of shortest path from node x to node y
$h(x)$	the height of node x ($h(x)$ represents just the height, not the unique height)
$hf(x)$	the full unique height of node x, $(h(x),x)$
$h(x).\alpha$	the first component of the height of node x
$h(x).\beta$	the second component of the height of node x

If the link (x, y) exists, then it is expressed as $(x, y) \in E$, otherwise $(x, y) \notin E$.

$$\forall\, x, y \in V \;\; x \in N(y) \Leftrightarrow y \in N(x) \Leftrightarrow (x, y) \in E \Leftrightarrow (y, x) \in E.$$

Since the graphs are undirected, nodes that are adjacent are adjacent in both directions. Also, saying

that a node is in the neighborhood of another node is equivalent to saying that there is an edge between them.

The system may evolve with time. When it is necessary, appending [*t*] to a symbol represents its value at time *t*, for example $E[t]$, or $h(x)[t]$.

Sequences are ordered sets and they are represented by listing their elements separated by commas in order enclosed by the characters "<" and ">". Certain set operations are defined for sequences. Membership, denoted with "∈" is a valid infix operator that indicates whether or not a particular symbol occurs in the sequence. Care must be taken in defining other set operations, such as union and disjunction are not clearly defined for sequences because they depend on the elements to be unordered. Sequences will have certain operations that are specific to them that are defined later.

Tuples are denoted by having their elements listed enclosed by "(" and ")" separated by commas, are not like sets or sequences. Tuples are a fixed length list of elements where like a sequence, the ordering of the elements has significance. Unlike in a sequence, however, each element in a tuple has a specific meaning associated with it and this meaning is assigned by the position within the list. In some cases, the meaning associated with each element in the tuple is the same, so the ordering effectively does not matter. We allow an abuse of notation in that tuples, when they are prefixes of other tuples may have fields appended to them and then become the other tuples.

TORA Specific Notation

$r(x)$		the reference level of node x
$h(x).\alpha$ $r(x).\alpha$	/	what TORA refers to as τ
$h(x).oid$ $r(x).oid$	/	the ID of the node originally defining x's reference level
$h(x).r$ $r(x).r$	/	reflected bit of x's height
$h(x).\beta$		what TORA refers to as δ

In the GB algorithms, α and β are used to represent characters in a height string. For notational consistency, α and β will also represent characters in the height strings of TORA. TORA usually refers to these terms as τ and δ, but they are conceptually equivalent to the α and β of the GB algorithms.

When referring to TORA, $h(x)$ is a 4-tuple defined as

$$h(x) \equiv (\alpha, oid, r, \beta).$$

$r(x)$, which applies only to TORA, and not to the general GB algorithms, is given by the 3-tuple

$$r(x) \equiv (\alpha, oid, r).$$

$r(x)$ is also called the reference level and serves as a convenient way to refer to the first three fields of $h(x)$.

1.3 *Link Reversal Algorithms*

TORA is based on a group of link reversal algorithms that we will refer to as the Gafni-Bertsekas (GB) algorithms [3]. The GB algorithms provide loop free routes in a network with bidirectional links to a single destination in the network using only information available locally, from adjacent nodes. GB algorithms, unlike other distance vector routing algorithms, such as distributed Bellman-Ford, do not suffer from routing table loops.

The algorithm assigns heights to each node such that the nodes can be totally ordered by their heights. This ordering on the nodes implies a direction to each of the links: the links are directed from nodes with greater heights to the nodes with the lower heights. This creates a directed acyclic graph (DAG) from the undirected graph.

The way that the algorithm assigns heights is by updating only those nodes that become local minima and therefore have no outgoing links. When a node other than the destination becomes a local minimum, that is all of its neighbors have heights that are greater than its own, it increases its height so that it is no longer a local minimum. As long as local minima other than the destination exist in the network, their heights continue to increase, until only the destination node is a local minimum. When this occurs, and all nodes except the destination have neighbors that are lower in height, no more events are enabled, assuming a fixed topology. The resulting height assignment is such that starting at any node in the network, by following links that lead to nodes of lower height, eventually the destination is reached. The paths will not form any loops because the heights of the nodes are totally ordered and the hops along the paths must proceed by strictly decreasing node height, guaranteeing uniqueness of the nodes traversed. For a proof of all the properties discussed here, see the paper by Gafni and Bertsekas.

There are assumptions that an algorithm must satisfy in order to guarantee the properties to be described below:

P1) The only time a node may update its height is when it assumes a greater height, reversing the

direction of its links when it is a local minimum. Decreasing height is forbidden. This rule has one exception: for the destination node, height updates are never allowed.

P2) The new height must depend solely on the heights of the neighbors of the node.

P3) An unbounded number of link reversals must lead to the height of the node becoming unbounded.

With these assumptions, additionally assuming that the network is not under partition (not partitioned meaning that all nodes are connected to the destination), the following properties apply.

By construction, the paths are always loop-free. However, the algorithm will exhibit routing loops while the heights are evolving and links are reversing directions. When a link reverses directions, packets that traversed the link just prior to the reversal now have an option of going backwards, up the same link that they just traversed. The routing loops formed in this fashion are purely transient and once the algorithm converges, all the routes are loop-free.

The algorithm always converges in a bounded period of time. The algorithm is also stable in that any node that has a directed path to the destination will not undergo any further reversals. Furthermore, the number of reversals and the final resulting heights depend only on the initial conditions of the network, though multiple paths (the algorithm behaves non-deterministically) can be used to reach the final state.

Like other distance vector algorithms, GB algorithms count to infinity under network partition. Since the heights are totally ordered, there will always be a globally minimal height, which implies that there will always be at least one locally minimal height. When the network is partitioned, that is the destination is not connected to the network, the local minimum cannot be the destination. Since there is always a local minimum that is not the destination, height updates are always enabled. The heights in the network increase indefinitely.

2. Proof of Correctness

TORA stands for Temporally-Oriented Routing Algorithm [8]. The temporally-oriented comes from the fact that TORA uses timestamps to create new heights. Using timestamps enhances performance over other GB algorithms. Another significant difference between TORA and the GB algorithms is that it does not suffer from the count to infinity problem under network partition. TORA includes a partition detection mechanism that takes advantage of the way height increases diffuse throughout a network.

2.1 *Not Gafni-Bertsekas*

As mentioned above, TORA uses timestamps for the new heights violating assumption (P2). This immediately puts TORA outside of the Gafni-Bertsekas class of algorithms. TORA does not have path independence. The set of final heights can vary, even for the same 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 always converge in a finite period of time. Establishing this formally is one of the primary goals of this work.

2.2 *Advantage of Temporally-Oriented Heights*

Since the new heights are based on time, 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 'oscillations' are necessary before all the heights converge.

2.3 *Partition Detection*

TORA includes a partition detection mechanism. Under certain conditions, it is possible for a partition to be detected when none actually exists. What the algorithm can guarantee is that if a partition is detected by TORA, then at some point in time previous to the partition being detected, a partition did occur, that is part of the network became completely disconnected from the destination. This result will be proved in a later section. There is a problem though, in that sometimes the network will become partitioned, but then another topology change may cause the network to become connected again. In this case, it is possible for TORA to detect a partition when it does not actually exist.

2.4 *TORA Model*

To improve the tractability of analyzing TORA, we omit modeling the details of the query response mechanism. We construct a simplified model of TORA similar to the model of link reversal algorithms presented by Gafni and Bertsekas[3].

In TORA, each node in V has a height associated with it. The heights and reference levels of nodes are ordered lexicographically, that is, they are equal only when all fields are equal and $h_1 > h_2$ if the first different field counting from left to right, of h_1 is greater than that of h_2. Each node has a unique identifier and these identifiers are totally ordered. This lexical ordering has the following obvious result for any nodes $x, y \in V$

$$r(x) > r(y) \Rightarrow h(x) > h(y) \Rightarrow hf(x) > hf(y).$$

TORA by definition adheres to (P1), that nodes may update their heights when they are local minima. Define S to be the set of nodes that are local minima excluding the destination.

$$S \equiv \{ x \in V \mid x \neq destination \wedge \forall y \in N(x) \; hf(y) > hf(x) \}. \quad (1)$$

This set is of interest because it is on this set that reversals are enabled. When S is empty, no further height update events are possible and the algorithm has converged, at least while the topology remains constant.

Lemma 1: $\forall (x, y) \in E \; \neg(x \in S \wedge y \in S)$. If x and y are adjacent, only one may be a local minimum.

Proof: Suppose $\exists (x, y) \in E \; (x \in S \wedge y \in S)$.

$$(x, y) \in E \Rightarrow y \in N(x) \quad (2.1)$$

$$x \in S \Rightarrow \forall z \in N(x) \; hf(z) > hf(x) \Rightarrow hf(y) > hf(x) \quad (2.2)$$

(By (1))

$$y \in S \Rightarrow \forall z \in N(y) \; hf(z) > hf(y) \Rightarrow hf(x) > hf(y) \quad (2.3)$$

(2.2) and (2.3) are direct contradictions of each other, so $\neg\exists (x, y) \in E \; (x \in S \wedge y \in S)$. □

Corollary 1: Link reversal events are only enabled for non-neighboring nodes.

Proof: By (P1) and *Lemma 1*. □

One of the features of this algorithm is that no assumption about the atomicity of events is necessary. This is because

Corollary **1** excludes any two adjacent nodes from both updating at the same time. This means there is no contention between neighbors performing height updates simultaneously. The algorithm does not even require that the updates occur in order. The algorithm, viewed at this level, only requires that the updates can be reliably sent between nodes. This assumes that the topology is fixed while the information is being updated. While topology changes may disrupt the operation of the algorithm, TORA is only guaranteed to converge while the topology remains constant.

We now formalize the update rule for nodes in S. Let $x \in S$. Let λ represent the event causing x to become a local minimum. Let t be the time at which the update takes place. Let h' express the new height to be selected. h' can also be expressed as components r' and β'. h' is selected according to the criterion below.

(1) If λ is a link failure
then $h' := (t, x, 0, 0)$. (3)

(2) If λ is not a link failure
and $\exists y, z \in N(x) \; r(y) \neq r(z)$ (4.1)
then let $r^* \equiv \max_{y \in N(x)} (r(y))$ (4.2)

and $h' := \min_{\{y \in N(x) | r(y) = r^*\}} (h(y)) + (0,0,0,-1)$. (4.3)

(3) If λ is not a link failure
and $\neg\exists y, z \in N(x) \; r(y) \neq r(z)$ (5.1)
(complement of 4.1)
and $\forall y \in N(x) \; r(y).r = 0$ (5.2)
then $r' := r(y) + (0, 0, 1)$ for any $y \in N(x)$
and $\beta' := 0$. (5.3)

(4) If λ is not a link failure
and $\neg\exists y, z \in N(x) \; r(y) \neq r(z)$ (6.1)
(same as 5.1)
and $\forall y \in N(x) \; r(y).r = 1$ (6.2)
(complement of 5.2)
and $\forall$ y $\in N(x) \; r(y).oid = x$ (6.3)
then a partition is detected.

(5) If λ is not a link failure
and $\neg\exists y, z \in N(x) \; r(y) \neq r(z)$ (7.1)
(same as 6.1)
and $\forall y \in N(x) \; r(y).r = 1$ (7.2)
(same as 6.2)
and $\forall$ y $\in$ N(x) $r(y).oid \neq x$ (7.3)
(complement of 6.3)
then $h' := (t, x, 0, 0)$. (7.4)
(same as (3))

Note that in case 4, no height is assigned because a partition is detected. Also note that we will not model the events that occur after the partition is detected and assume that TORA's CLR flood works properly.

2.5 *TORA Properties*

Using the above formalisms for TORA, we shall prove that for a connected, static topology, TORA converges in a finite number of steps.

Lemma 2: Whenever a node increases its height in a reversal, its reference level increases.

Proof: Let x be a local minimum. Let r denote $r(x)$ while x is a local minimum, and let r' denote the reference level TORA chooses as the next reference level. Proceed by verifying the result, $r' > r$, for all cases.

Case (1) and (5): x generates a new globally highest reference level. The desired condition,

$$r' > r,$$

holds true trivially.

Case (2): This case applies only when

$$\exists\, y, z \in N(x) \;\; r(y) \neq r(z). \qquad (4.1)$$

TORA will choose to propagate the highest reference level of those nodes in $N(x)$. Let

$$r^* \equiv \max_{y \in N(x)} (r(y)).$$

Given (4.1), and the fact that r^* is a maximum,

$$\exists\, y \in N(x) \;\; r^* > r(y). \qquad (8)$$

Let y^* be any element of $N(x)$ satisfying (8). Since x is a local minimum, $r(y^*) \geq r(x)$, and

$$r^* > r(y^*) \geq r(x). \qquad (9)$$

When x takes r^* as its reference level, its reference level increases.

Case (3): Reflect back a higher sublevel when all neighbors have the same reference level. Since x is a local minimum, $\forall\, y \in N(x)\;\; r(y) \geq r(x)$. When reflecting back a higher sublevel, the new reference level $r' > r(y) \geq r(x)$.

Case (4): This case ultimately causes TORA to halt and clears the heights altogether. It does not really perform a reversal.

Since the statement holds true for all cases, it must be true. □

Lemma 3: Whenever a node increases its height in a reversal, its reference level becomes greater than or equal to the reference level of its highest neighbor.

Proof: Let x be a local minimum. Let r denote $r(x)$ while x is a local minimum and let r' denote the reference level TORA chooses as the next reference level for x.

Case (1) and (5): x generates a globally highest reference level. Obviously r' > the reference level of any node in $N(x)$.

Case (2): By (4.2) $r' \geq$ the reference level of all neighbors of x.

Case (3): All neighbors of x have the same reference level, s by (5.1).

$$r' = s + (0, 0, 1) > s.$$

Case (4): This does not cause a reversal to be performed.

In all cases, the result holds. □

Lemma 4: A node may perform at most two reversals until all of its neighbors reverse and increase their reference levels.

Consider case by case what happens after the first reversal. Let h be the height of x prior to its reversal and let h' be the height TORA selects to update x, and r' be the corresponding reference level.

Case (1) and Case (5): x generates a globally highest reference level $h' > h(y)\;\forall\, y \in V$. In order for x to become a local minimum again, all of its neighbors must increase in height to be higher than x. In these cases, x may only reverse again after all of its neighbors have increased.

Case (2): Since x performs only a partial reversal, it is possible that x is still lower than some of its neighbors after it reverses. Let

$$O \equiv \{\, y \in N(x) \mid hf(y) > hf' \,\}$$

be the set of x's neighbors that are still higher than x after its first reversal. x may become a local minimum again without these nodes reversing, though all of x's other neighbors must reverse before x may reverse. By (4.3), we know that

$$\forall\, y \in O \;\; r(y) = r'. \qquad (10)$$

Now consider x's second reversal. Let r'' be x's reference level after it reverses the second time. By *Lemma 2*, we know that $r'' > r'$ and by (10) we know that $\forall\, y \in O \;\; r'' > r(y)$. The only way that x may become a local minimum again to reverse for the third time is if all the nodes in O reverse, thus proving the result for case 2.

Case (3): Since prior to x's reversal, all of its neighbors are at the same reference level, and x takes on a reference level higher than its neighbors' reference level, in order for x to become a local minimum again, all of its neighbors must increase in height first. □

Corollary 2: For any two nodes in a connected, fixed topology graph, where Δ is the number of hops along the shortest path between the two nodes, the difference in the number of reversals between the two nodes must be less than or equal to 2Δ.

Proof: Proceed by induction on the number of hops, Δ. For $\Delta = 1$, *Lemma 4* states directly that the number of reversals can differ by at most 2, which equals 2Δ. Assume that for Δ - 1 hops, the result is true. Let $x, y \in V$ be two nodes that are Δ hops apart.

$$\exists\, z \in V \;\; z \in N(x) \wedge y \text{ and } z \text{ are } \Delta - 1 \text{ hops apart.}$$

By the inductive assumption, z can differ in reversal count with y by only by $2(\Delta - 1)$. Since $z \in N(x)$, and using *Lemma 4*, x may only differ in hop count from z by 2, so it can only differ in hop count from y by $2(\Delta - 1) + 2 = 2\Delta$. □

Theorem 1: *Convergence:* TORA always converges in a connected network. For a connected, fixed topology graph, TORA either converges in a finite number of steps, or a partition is detected.

Proof: Suppose that there exists some connected network for which TORA never converges and never detects any partitions. This implies that S is never empty and that there are an infinite number of reversal events. For this to be true, there must be at least one node x, that undergoes an infinite number of reversals. Let D be the diameter of the network (the length of the longest shortest path). By *Corollary 2* and the fact that the destination never reverses its height, we know that the upper bound on the number of reversals that any node can undergo is $2D$. This is a contradiction and thus TORA always converges in a finite period of time. This result is only valid when the network is connected to the destination. □

We have completed the proof that TORA always converges or detects a partition when the nodes in the network are connected to the destination. It is also possible to show that under certain conditions, TORA cannot detect a partition in a connected network.

Lemma 5: Let nodes $x, y \in V$, be s.t. $(x, y) \in E[t_0]$ and $hf(y)[t_0] < hf(x)[t_0]$. If $\exists\ t_1 > t_0\ \ hf(y)[t_1] < hf(x)[t_0] \wedge \forall\ \tau \in [t_0, t_1] (x, y) \in E(\tau)$ (11)

then $\forall\ \tau \in [t_0, t_1]\ \ hf(x)[\tau] = hf(x)[t_0]$.

In other words, given two nodes x and y that are initially adjacent and remain connected over the period of interest, if initially, $hf(y) < hf(x)$, and $hf(y)$ remains less than $hf(x)$, then $hf(x)$ must remain constant.

Proof: Assume (11) holds. By (P1), the fact that heights are non-decreasing,

$hf(y)[t_1] < hf(x)[t_0] \Rightarrow \forall\ \tau \in [t_0, t_1]\ \ hf(x)[t_0] > hf(y)[t_1] \geq hf(y)[\tau]$.

$\forall\ \tau \in [t_0, t_1]\ \ hf(x)[\tau] \geq hf(x)[t_0]$. (by (P1))

$\forall\ \tau \in [t_0, t_1]\ \ hf(x)[\tau] \geq hf(x)[t_0] > hf(y)[\tau]$.

$\forall\ \tau \in [t_0, t_1]\ \ \exists\ y \in N(x)\ \ hf(x) > hf(y)$.

Hence x cannot change its height because it is never a local minimum. Therefore x's height must be constant over $[t_0, t_1]$. □

Corollary 3: Let $X \equiv \{x_1, \ldots, x_n\} \subseteq V$ be such that the set

$P \equiv \{ (x, y) \mid \exists\ i \in \{1, \ldots, n\text{-}1\}\ \ x = x_i \wedge y = x_{i+1} \}$

is a subset of E and $\forall\ i \in \{ 1, \ldots, n\text{-}1 \}\ \ hf(x_i)[t_0] > hf(x_{i+1})[t_0]$.

If $\exists\ t_1 > t_0\ \ hf(x_n)[t_1] = hf(x_n)[t_0] \wedge \forall\ \tau \in [t_0, t_1]\ \ P \subseteq E(\tau)$ (12)

then $\forall\ x \in X\ \ \forall\ \tau \in [t_0, t_1]\ \ h(x)[\tau] = h(x)[t_0]$.

Proof: Proceed by induction on n applying *Lemma 5*. For $X \equiv \{x_1, x_2\}$, the result is a direct consequence of *Lemma 5*. Assume that the result holds true for $X \equiv \{x_1, \ldots, x_{n-1}\}$. In the case of $X \equiv \{x_1, \ldots, x_n\}$, assume that (12) holds. Then from *Lemma 5* we know that

$\forall\ \tau \in [t_0, t_1]\ \ h(x_{n-1})[\tau] = h(x_{n-1})[t_0]$.

Using the inductive hypothesis, the result follows. □

Corollary 3 generalizes *Lemma 5* to apply to chains of connected nodes. The result could be extended to have the same condition as *Lemma 5*, but it is not necessary.

The following is an important result that characterizes the propagation of reference levels throughout the network.

Lemma 6: Let x generate at time t_0, a new reference level $r \equiv (t_0, s, 0)$ where

$h(x)[t_0] = (t_0, x, 0, 0)$.

Assume that $\exists\ t_1 > t_0\ \ \forall\ \tau \in [t_0, t_1]\ \ E[t_0] = E[\tau]$, that is the topology is static over $[t_0, t_1]$. Then

$\exists\ y \in V\ \ r(y)[t_1] = r \Rightarrow h(x)[t_1] = h(x)[t_0]$.

Proof: r is uniquely generated by node x at time t_0. Let $y_1 \in V$ be any node such that $r(y_1)[t_1] = r$. This reference level may be reached in only two ways. All node IDs being unique guarantees that x is the only node that may generate r. If y_1 is any node other than x, then it must take on this reference level does so by propagation, which is by case (2) of the height selection algorithm.

Since prior to t_0, reference level r does not exist, and $r(y_1)[t_1] = r$, $\exists\ t \in [t_0, t_1]$ where y_1 reverses and takes on reference level r. By *Lemma 2*, y_1 may only update its reference level to r once, so t is unique. The condition below is necessary for the propagation of reference level r at time t to node y_1.

$(\forall\ z \in N(y_1)\ \ hf(z)[t] > hf(y_1)[t])$ (13)

$\wedge\ (\exists\ m \in N(y_1)((\forall z \in N(y_1)\ (m = z \vee hf(m)[t] > hf(z)[t]))$ (14)

$\wedge\ \exists\ z \in N(y_1)\ r(m)[t] > r(z)[t])$. (15)

(13) states that y_1 must be a local minimum. (14)-(15) state that there must exist some neighbor m where its height is greater than any of y_1's neighbors and there must be another neighbor of y_1 with a reference level strictly less than $r(m)[t]$. Let y_2 be the node satisfying conditions (14) and (15). When y_1 updates its height at time t, according to (4.3), $h(y_1)[t] < h(y_2)[t]$ and $h(y_1)[t_1] = h(y_1)[t] < h(y_2)[t]$ so by *Lemma 5*, $h(y_2)[t_1] = h(y_2)[t]$. The statement below summarizes the result.

$$\forall\, y_1 \in V \;\; r(y_1)[t_1] = r \Rightarrow y_1 = x \vee \exists\, y_2 \in N(y_1)\; r(y_2)[t_1] = r \wedge h(y_2)[t_1] > h(y_1)[t_1] \quad (16)$$

Equation (16) has a recursive structure. Assume that $y_1 \neq x$. Then there exists y_2 with reference level r, having a height at t_1 strictly greater than $h(y_1)[t_1]$. Now y_2 is another node, where the condition $r(y_2)[t_1] = r$ holds. If y_2 is not x, by (16) again, there exists another node y_3 such that $r(y_3)[t_1] = r$. Also, $h(y_3)[t_1] > h(y_2)[t_1] > h(y_1)[t_1]$. This recursion can be repeated whenever the next node discovered is not x. Any sequence $<y_1, \ldots, y_n>$ generated in this way is always increasing in height, so each node in the sequence is unique. Since there are a finite number of nodes in the network, the recursion must terminate and the only way it can terminate is if $y_n = x$. □

Corollary 4: Under the same conditions given in *Lemma 6*,

$$\forall\, y \in \{ y \in V \mid r(y)[t_1] = r \wedge y \neq x \} \quad \exists\, X \equiv \{x_1, \ldots, x_n\} \subseteq V$$
$$\forall\, x \in X \;\; r(x) = r$$
$$\wedge\; x_1 = y \;\wedge\; x_n = x$$
$$\wedge\, \forall\, i \in \{ 1, \ldots, n\text{-}1 \} \quad (x_i, x_{i+1}) \in E \wedge h(x_i)[t_1] < h(x_{i+1})[t_1].$$

In other words, starting at y, there exists a path of connected nodes connected, such that all have reference level r, and increase in height, terminating at x.

Proof: Follows in arguments given in *Lemma 6*. □

Lemma 6 and *Corollary 4* illustrate how a reference level propagates through the network. As a newly defined reference level propagates, a DAG is formed, rooted at the node generating the new reference level. This DAG consists of nodes all having the same reference level. In order for the root node to become a local minimum and reverse, it is necessary that no nodes in the entire network have the same reference level.

Lemma 7: Assume that the topology is fixed. Let $x \in V$ generate a new reference level r at time t_0. Assume that x detects a partition at time t_1 by having its reference level reflected back.

$$\forall\, \tau_1 \in [t_0, t_1] \;\; \forall\, y \in \{ y \in V \mid y \neq x \wedge r(y)[\tau_1] = r \} \quad \exists\, \tau_2 \in (\tau_1, t_1) \;\; \forall\, t < [\tau_1, \tau_2] \;\; r(y)[t] = r \wedge r(y)[\tau_2] = r + (0, 0, 1).$$

In other words, any y that acquires reference level r, must update its reference level from r to the reflected reference level $r + (0, 0, 1)$ in order for x to detect a partition.

Proof: Assume that that the topology is fixed and node $x \in V$ generates a new reference level r at time t_0. At time τ_1, let $y \neq x \in V$ be such that $r(y)[\tau_1] = r$. By *Corollary 4*,

$$\exists\, X \equiv \{ x_1, \ldots, x_n \} \;\forall\, x \in X \;\; r(x) = r \wedge x_1 = y \wedge x_n = x \wedge \forall\, i \in \{ 1, \ldots, n\text{-}1 \} \;\; (x_i, x_{i+1}) \in E \wedge h(x_i)[t_1] < h(x_{i+1})[t_1].$$

For this set X, we proceed by induction on n to show for $n > 1$, the result is true. For $n = 2$, where $y \in N(x)$, we know by the fact that x detects a partition at time t_1, and by required conditions (6.1)-(6.3), that $r(y)[t_1] = r + (0, 0, 1)$. Since $r(y)[\tau_1] = r$, the result follows by (P1).

Now assume that the result holds for n-1. Let $y \in V$ be such that $r(y)[\tau_1] = r$ and the set X associated with y by Corollary 4 satisfy $|X| = n$. Then $x_2 \in N(y)$ satisfies the criterion for case n-1. By the inductive hypothesis, x_2 must change reference level from r to $r + (0, 0, 1)$. This means that x_2 must become a local minimum. This cannot happen until x_1 increases its reference level so that $hf(x_1) > hf(x_2)$. Let $r' > r$ be the reference level that x_1 increases to. Suppose that $r' \neq r + (0, 0, 1)$. Since $r + (0, 0, 1)$ is the least reference level greater than r. Then $r' \neq r + (0, 0, 1) \Rightarrow r' > r + (0, 0, 1)$ by *Lemma 2*. This means that when node x_2 updates its height, its reference level will be at least r'. This contradicts the inductive hypothesis, so $r' = r + (0, 0, 1)$. □

Corollary 5: Assume the topology is fixed. Let $x \in V$ generate a new reference level r at time t_0. If at any time before x detects a partition,

$$\exists\, y \in V \;\; r(y) = r \wedge \exists\, z \in N(y) \;\; r(z) > r + (0,0,1) \quad (17)$$

then x_1 cannot detect a partition.

Proof: By *Lemma 7*, all nodes acquiring reference level r must reverse to reference level $r + (0, 0, 1)$. *By Lemma 3*, any node reversing, must take on a reference at least as high as its highest neighbor, which in this case has a reference level greater $r + (0,0,1)$. □

Lemma 8: Assume that the topology is fixed and all nodes are connected to the destination d. Let $x \in V$ generate a new reference level r at time t_0. For any node that propagates the reference level $r + (0, 0, 1)$ via case (2) of the decision tree, two conditions hold at the time when it updates its reference level to $r + (0,0,1)$.

1) All its neighbors have reference level r or the reflected reference level $r + (0, 0, 1)$.
2) It must have reference level r.

Proof: Let z be a node generating the reflected reference level $r + (0, 0, 1)$ by case (3) at time $t > t_0$. Proceed by induction using the neighbors of z as the base case.

Let w be any node in $N(z)$. Case (3) requires that all neighbors of z have reference level r, so $r(n)[t] = r$. Assume that w takes on the reflected reference level at time $t_1 > t$. By *Lemma 2*, the fact that $r(n)[t] = r$, and the fact that $r + (0, 0, 1)$ is the minimum reference level greater than r, $\forall\ \tau \in [t, t_1]\ \ r(n)[\tau] = r$, so the second property is true for w. In order for w to be a local minimum

$\forall\ v \in N(w)\ \ hf(v)[t_1\text{-}] \geq hf(w)[t_1\text{-}]$.

$\forall\ v \in N(w)\ \ r(v)[t_1\text{-}] \geq r$.

$\forall\ v \in N(w)\ \ r + (0, 0, 1) \geq r(v)[t_1\text{-}]$.

(if not, w propagates a higher reference level)

$\forall\ v \in N(w)\ \ r(v)[t_1\text{-}] = r$.

So the first property is true for w. The result holds for all neighbors of any node generating the reflected reference level.

Assume that the first and second property hold for some node $w \in V$. Assume that w takes on the reflected reference level at time $t_1 > t$. Let $v \in N(w)$ $r(v)[t_1] \neq r + (0, 0, 1)$. Assume that v takes on reference level $r + (0, 0, 1)$ at time $t_2 > t_1$. By the first property applied to node w, $r(v) = r$. Therefore, by the argument for the base case, the second property applies to node v. Arguing again as in the base case, the first property must also apply to node v. The result holds for the neighbor of any node for which the result holds.

Since the only way a node can propagate reference level $r + (0, 0, 1)$ is by

a) being a neighbor of a node generating the reflected reference level or
b) being a neighbor of a node propagating the reflected reference level,

it must be true in all cases. □

Definition: The *frontier* of r, where r is a reference level, denoted $f(r) \subseteq V$, is defined

$f(r) \equiv \{\, y \in V \mid r(y) \neq r \wedge \exists\ z \in N(y)\ \ r(z) = r \,\}$.

Theorem 2: *Correctness Criterion: No partitions detected in connected networks*

Assume that the topology is fixed and all nodes are connected to the destination d. Let $x \in V$ generate a new reference level r at time t_0. x cannot detect a network partition through case (4) of the height selection process.

Proof: Assume that the topology is fixed and all nodes are connected to the destination d. Let $x \in V$ generate a new reference level r at time t_0. Since all nodes are connected to d, then there must be a path from x to d.

Let $F \equiv f(r)$.

Let $M \equiv \{\, y \in V \mid r(y) = r \,\}$.

Let $G \equiv \{\, y \in M \mid \forall\ z \in M\ \ D(z, d) \geq D(y, d) \,\}$.

In other words G is the set of nodes having reference level r with the shortest distance to the destination. F, M and G change with time so let $F[t]$, $M[t]$ and $G[t]$ denote their respective values at time t.

Proceed by showing that in all cases either

1) $\forall\ t > t_0\ \ |G[t]| > 0 \wedge \forall\ \tau_1 \in [t_0, t)\ \ \forall\ \tau_2 \in (\tau_1, t]\ \ \forall\ y \in G[\tau_1]\ \ \forall\ z \in G[\tau_2]\ \ D(y, d) \geq D(z, d)$. That is $|G| > 0$ and the distance between G and the destination is non-increasing with time. Since $|G| > 0$ and $\forall\ y \in G\ \ r(y) = r$, by *Lemma 6*, x cannot detect a partition.

or

2) $\exists\ t > t_0, \exists\ y \in F[t]\ \ r(y)[t] > r + (0, 0, 1)$. Then by *Corollary 5*, x cannot detect a partition.

Initially, the first condition is satisfied. $|G| = 1 > 0$ and since there are no comparisons, the distance between nodes in G and d satisfy the non-increasing criterion. It is possible for the second condition to hold also, and either way, the property holds true.

Assume that the first condition is satisfied. Now proceed by showing that for all enabled events, either the first or second result will hold. If this is the case, then no sequence of events may ever cause the conditions to be violated and the proof is complete.

There are only two events that may directly affect G.

A. Node y with reference level $r(y) \neq r$ updates its reference level to $r(y) = r$.
B. Node $y \in G$ updates its reference level from $r(y) = r$ to $r(y) > r$.

Event A can only occur for nodes in F by definition. Event B applies to nodes in G.

Consider the effects of event A on conditions 1 and 2. Let $y \in F[t]$ be a local minimum at time t. Assume that conditions 1 and 2 hold prior to updating the height of y to r. If

$$\forall\ z \in G[t]\ \ D(z, d) > D(y, d) \qquad (18)$$

then $G[t]$ will be replaced with $\{\, y \,\}$ after the event occurs. Since y is closer to the destination than any node in $G[t]$, condition 1 is preserved. If

$$\forall\ z \in G\,[t]\ \ D(z, d) = D(y, d), \qquad (19)$$

then $G[t]$ will be replaced with $G[t] \cup \{\, y \,\}$, still preserving condition 1. Otherwise, if

$$\forall\ z \in G[t]\ \ D(z, d) < D(y, d), \qquad (20)$$

$G[t]$ is unaffected and condition 1 is still preserved. Any occurrence of event A preserves condition 1. Note that once $\forall\ z \in G\ \ D(z, d) = 1$, the only frontier

node is the destination, but since the destination cannot update its height, (18) is no longer reachable.

Consider now the effects of event B on conditions 1 and 2. Let $y \in G$ be a local minimum at time t. Let $N \equiv F \cap N(y)$. $y \in G \Rightarrow$

$$\exists\, z \in N(y)\ \ r(z)[t] = r. \quad (21) \quad \text{(by } \textit{Corollary 4}\text{)}$$

$$\exists\, z \in N\ \ r(z)[t] \neq r\ \wedge\ D(z, d) = D(y, z) - 1. \quad (22)$$

(by definition of G)

Since y is a local minimum and by the fact that $\forall\, z \in F\ \ r(z) \neq r$,

$$\forall\, z \in N\ \ r(z) > r. \quad (23)$$

$$\Rightarrow\ \forall\, z \in N\ \ r(z) \geq r + (0, 0, 1). \quad (24)$$

Assume $\exists\, z \in N\ \ r(z) > r + (0, 0, 1)$, condition 2 is satisfied and the result holds. Otherwise suppose

$$\neg\exists\, z \in N\ \ r(z) > r + (0, 0, 1) \quad (25)$$

$$\text{then } \forall\, z \in N\ \ r(z) = r + (0, 0, 1). \quad (26)$$

(with (24))

By *Lemma 8* and (26) $\forall\, z \in N$ z must have had reference level r just prior to having reference level $r + (0, 0, 1)$. By (19), $\exists\, z \in N\ \ D(z, d) < D(y, z)$. This contradicts the assumption that condition 1 holds prior to event B occurring: the distance between nodes in G and the destination are non-increasing with time. Therefore, (25) is not reachable.

The conditions are satisfied by all possible events. □

Theorem 2 shows that any node generating a new reference level cannot detect a partition if the topology remains static after the reference level has been created. However, in cases where the topology is dynamic and changing, it is easy to produce cases of partitions being detected in connected topologies.

There are two distinct cases of partition detection when the topology is allowed to change. In one case, partitions have never existed in the network and the fact that a topology change can lead to a partition being detected is an artifact of the algorithm. In the other case, a network partition existed transiently and was detected, but is already in the process of communicating the partition being detected. The first case is avoidable, but the second case is not in the current framework.

3. Conclusions and Suggestions for Future Research.

We have developed a formal proof method for the correctness of TORA, a dynamic adaptive routing algorithm for mobile ad-hoc networks (MANETs). We have also developed a formal proof of a partition detection criterion. While all tests have failed to find anything wrong with the algorithm, there is still no guarantee that it is correct in its full specification, as our proof requires simplifications and assumptions to be placed on the algorithm.

The ultimate goal of this type of research is to develop a way to automatically check a specification for correctness and liveness properties. This is currently an entirely open ended question. While our results do verify correctness in TORA, it is only under certain assumptions that the proof is valid. The vision of being able to go automatically from a specification to a proof of correctness still requires further research and progress.

References

1. S. Bayern. 2001. Synchronized recursion. *Dr. Dobb's Journal* (June) : 151-155.
2. G. Chartrand and L. Lesniak. 1996. *Graphs and Digraphs*. London: Chapman & Hall.
3. E. Gafni and D. Bertsekas. 1981. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications* 29 (January) : 11-18.
4. E. Gamma, et al. 1995. *Design Patterns*. Reading: Addison-Wesley.
5. G.J. Holzmann. 1997. The model checker Spin. *IEEE Transactions on Software Engineering* (May) : 279-295.
6. L. Lamport. 1994. How to write a long formula. *Formal Aspects of Computing Journal* (September) : 580-584.
7. B. McKay. 1981. Practical graph isomorphism. *Congress Numerantium* (30) : 45-87.
8. V. Park and M. Corson. 1997. A highly adaptive distributed routing algorithm for mobile wireless networks. *IEEE Proceedings of INFOCOM* (April) : 1405-1413.
9. R. Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM Journal of Computing* (1) : 146-160.
10. S. Yang, *TORA, Correctness, Proofs and Model Checking,* M.S. Thesis, Electrical and Computer Engineering Department, University of Maryland, College Park, December 2002.