Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 119 (2005) 67–92
www.elsevier.com/locate/entcs

Towards A Game Theoretic Understanding of Ad-Hoc Routing

Irfan Zakiuddin[a,1] Tim Hawkins[b,2] Nick Moffat[b,3]

[a] *Command and Intelligence Systems, QinetiQ, Malvern, UK*

[b] *Trusted Information Management, QinetiQ, Malvern, UK*

Abstract

In this paper we present a new application of game theory, in which game theoretic techniques are used to provide a rigorous underpinning to the analysis of ad-hoc routing protocols. The explosion of interest in ad-hoc networks over the last few years has resulted in a very large number of routing protocols being proposed. Despite this, the science of analysing routing protocols is still relatively immature, and the question that remains is how to decide "how good" a given protocol is. We propose a game theoretic approach as a potentially effective means of answering this question. The conceptual mapping of routing into a game is, we believe, natural and simple. Furthermore, game theory provides an extensive repertoire of tools to analyse key properties. The paper describes how routing techniques can be modelled as games and presents some analytical results.

Keywords: game theory, routing, ad-hoc networks, performance analysis

1 Introduction

Game theory comprises a powerful set of techniques to reason about situations involving conflict and competition. The subject of this paper is the application of game theoretic techniques to the analysis of ad-hoc routing protocols. We believe this to be a novel use of such techniques. The particular way in which we have applied game theory to this problem is described in Section 3.

[1] Email: I.Zakiuddin@signal.QinetiQ.com
[2] Email: T.Hawkins@eris.QinetiQ.com
[3] Email: N.Moffat@eris.QinetiQ.com

doi:10.1016/j.entcs.2004.07.009

However, we begin with an overview of the current state of research into the development and analysis of ad-hoc routing protocols.

Over the last few years there has been an explosion of interest in ad-hoc networking research [1]. A good portion of this research has aimed to develop routing protocols for ad-hoc networks. The result is that there are now very many ad-hoc routing protocols. Part of the motivation for designing new protocols comes from wanting to meet the challenging requirements of the ad-hoc networking domain; some of these challenges are discussed below. Clearly, designers of new protocols, or developers of ad-hoc networks, need to analyse protocols to determine how well requirements are met.

The primary analytical technique is based on simulation, a number of simulation tools are in use such as OPNET Modeller [2], NS-2 [3] and GloMoSim [4]. However, these tools do not appear to provide consistent results [5]. Furthermore, [6] and [7] find a discrepancy between the results of simulation and field trials deploying actual ad-hoc networks. The question may be asked: why does simulation give inconsistent and unreliable results? Two answers come to mind:

- A simulator attempts to represent reality, and its predictions fail because its representation of reality is not sufficiently faithful.
- When a model's behaviours are sampled and analysed by the simulator, the sample chosen is too small and not sufficiently representative of the behaviour of the protocol. In this regard note that the work on mobility models has the effect of determining, up front, a subspace of the full sample space [8].

When the cited papers speculate on the reasons for the various poor simulation results that they find, their reasons fall in either (or both) of the categories above.

Clearly, if the model is perfectly faithful and the sample covers the full space of the model's behaviour, then the results of the analysis will be both consistent and correct. The requirement of perfect fidelity is, of course, impossible, while that of full coverage is at best very difficult, even for models that are very abstract and restricted in the size of the network model. Nevertheless, these two principles, namely *fidelity* and *coverage* provide the basis and the inspiration for our work, which attempts to take a fresh look at the problem of analysing ad-hoc routing protocols.

1.1 Our Approach and This Paper

In our approach we aim to start with abstract models which we analyse rigorously, and then to use the results of these analyses as the basis for a guided

heuristic search over more accurate models. In effect we intend to combine the benefits of high coverage (on low fidelity models) with high fidelity models (analysed with limited coverage).

This paper concerns the first step in this programme: how we are analysing low fidelity models rigorously. Section 2 discusses the level of abstraction at which we model routing protocols. That section also discusses the first approach that comes to mind for high coverage analysis of the protocol, namely model-checking, and why we chose not to use model-checking. We believe that game theory provides an approach to analysing routing protocols that is both natural and powerful. In Section 3 we give a very brief description of game theory and how it can be used to analyse routing protocols. Section 4 highlights some generic issues that arise when mapping routing protocols to games. The results we have achieved to date with game theory are presented in Sections 5, 6 and 7. This is still an early phase of our proposed programme and there are various limitations to our current game theoretic analysis technique; we conclude in Section 8 with a discussion of what these limitations are, how they may be addressed and the way ahead.

2 Rigorous analysis of routing techniques

2.1 Routing Techniques

Recent research has lead not only to many routing protocols, but to many *routing techniques*. By a routing technique we mean the basic strategy that a routing protocol uses to store and propagate routing information. A routing technique captures the following three protocol characteristics:

- whether routing information is propagated on demand or proactively;
- whether routing information is propagated by flooding or by propagation on a spanning tree; and
- the format of the routing information stored locally and communicated (whether it is link state data, distance vectors, or whatever).

Thus examples of routing techniques might be: proactive flooded distance vector, proactive multicast link state routing, reactive flooded link reversal, and so forth.

Routing techniques can be regarded as abstractions of specific routing protocols; clearly there are several protocols for each technique. Furthermore, it is possible to model just the routing technique, rather than a protocol instance of a technique (we discuss this in Section 3). In our opinion the most abstract level at which a routing protocol can usefully be modelled is that of the routing technique it instantiates.

Having fixed on routing techniques as the level at which we want to begin our investigation, the obvious questions are: what do we want to learn about the techniques? And, what analytical tools do we intend to use?

2.2 What Do We Want to Learn?

We can divide routing protocol properties into two sorts:

(i) *Soundness.* This means that routers make correct routing decisions – but this property is insensitive to the delay involved, or the resources consumed, in order to make that decision. Soundness amounts to saying that the protocol is guaranteed to terminate, eventually and after an unspecified amount of communication, with the routers having the correct view of the network topology.

(ii) *Performance.* These are also known as efficiency requirements. We will concentrate on the following:
- *convergence*, which is a measure of how quickly a routing protocol responds to changes in the network topology,
- *network overheads*, which is a measure of the network resources that a routing protocol consumes, simply to enable it to make correct routing decisions.

The subject of soundness is of interest to routing protocol designers (the ARPANET bug [9] makes soundness impossible to ignore!), but most of the effort, on the part of engineers and ad-hoc protocol researchers, goes into achieving and improving performance. Hence both convergence and network overheads are well known, and standard texts that cover routing (such as [9]) discuss them in detail. However, we have not found in the literature a clear analysis of how the various routing techniques compare with respect to these two performance characteristics. A key objective of our work, the first stage of which is reported in this paper, is to understand this issue.

We can study both the soundness and the performance of routing techniques, and we believe that this makes a good starting point for our programme of providing a rigorous underpinning to the study of routing protocols. In fact we hope that rigorous analysis of the performance of routing techniques will provide a firm basis for understanding important questions, such as:

(i) What is the impact on convergence of propagating routing information on a spanning tree, as opposed to flooding the routing information?

(ii) How much reduction is there to the network overheads as a result of using spanning tree methods?

(iii) Do reactive routing approaches consume less network resources than

proactive ones?

The received wisdom on the last point is that reactive approaches do reduce overheads, but we have heard this belief questioned, and we are not aware of an exact basis for this belief. Having fixed what we want to learn about routing techniques, the next decision to make is what tools to use.

2.3 What Tools Can We Use?

We are aiming for high coverage in our exploration of the behaviours of a model of a routing technique. Over the last ten years, the use of model-checking [10,11] has increased substantially, essentially as a branch of formal methods, based on the results it has been able to deliver from its exhaustive analysis of models. The primary use of model-checking is to verify that a design is correct by creating a model of that design and then checking the complete behaviour of the model against a specified property. Model-checkers mainly verify *safety* and *liveness* properties [12] of which the soundness properties of routing are an instance. However, there are model-checking tools that analyse models with timing and probabilistic behaviour [13]; could these be used to study performance of routing techniques?

We believe it is unlikely that probabilistic model-checkers will provide an effective technology for analysing routing protocol performance; rough calculations on the size of the space of behaviours (usually called the *state space*) that the model-checking tool must check shows why. In a model of 5 nodes, with each node a router, there are 10 links, under the simplifying assumption that all links are bi-directional. To model the technique of proactive link state unicast, each router will need to retain at least one piece of information for each link, *viz.* whether it is up or down. This means that each router in the model has at least 2^{10} states (two states for each of 10 links), so a 5 node model has at least $(2^{10})^5$ or 2^{50} states. This simple calculation assumes that the routers are able to learn about, or become confused about, network topology independently of each other. [4]

A model of 2^{50} states is already intractable for analysis by typical computing engines, but that is not the end of the story. Such a model would only be able to study soundness, since it only has binary information about whether links are up or down. To study performance we need to do two things:

(i) Add information about time, communication delays, rate of change of the topology, link quality, etc.

[4] In reality, there may be some correlation between the states of different routers, but on the other hand 5 nodes is not many!

(ii) Change the tool from a safety-liveness verifier to a time and probablity model-checker.

Both these changes are likely to increase the state space dramatically. A typical response to this sort of problem, from the formal verification community, is that abstraction techniques need to be used. The use of abstraction to make intractable safety-liveness verification problems tractable is interesting, useful and well-established. However, the use of analogous techniques, in the domain of performance analysis with model-checkers contains many deeper and harder problems.

For the reasons given above, we are skeptical that model-checking has the capability to meet our needs. Nevertheless, it may be that models of 4 routers can be analysed by model-checkers for performance properties, and that such analysis can be used to validate the approach that we have chosen, which we present next.

3 Modeling a routing protocol as a game

Game theory [14] was invented to provide a mathematical foundation for reasoning about conflict and competition. It has grown into a rich theory, with powerful mathematical and computational tools. It also has the advantage of retaining its intuitive appeal, which is what first attracted us to it. Two insights enabled us to turn this pool of theory and tools into a potentially powerful analytical capability for analysing routing:

(i) A routing protocol can be modelled as a minimax game between the network and the routers.

(ii) The minimax value of the game can quantify the performance properties.

To explain how these two insights are the basis for using minimax game theory to analyse routing, we first need a brief summary of the game theory that we will use – standard texts [15,16] discuss the subject more comprehensively.

3.1 Basic Form of a Minimax Game

In a minimax game each player chooses game moves to maximise his or her guaranteed minimum gain (hence the term *minimax*); in simple language, players attempt to make the bottom line as high as possible.

Suppose we are considering a two-player game in which, at each turn, a player has a choice of three legal moves. Figure 1 might represent possible choices for the first two moves of such a game; it is a tree in which each node represents a state of the game and each branch represents a legal move

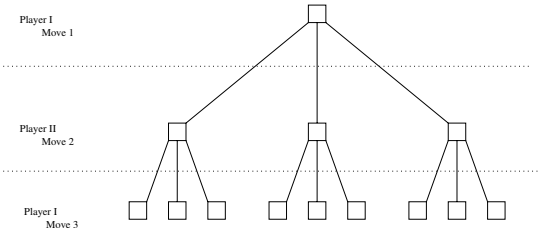

Fig. 1. A representation of a game

between two states. Such a structure is called a *game tree*.

A *run* of the game is a path through the game tree starting at the root node (which represents the initial state) and ending at a leaf node (at which the game has ended). Each run of the game therefore corresponds to a particular sequence of moves chosen by the two players in turn. A *cost function* is evaluated at the leaves of the game tree. It maps each complete run of the game to a value representing the outcome of the game for this run. The outcome is the cost of this run for a particular player, known as the *minimising player*. The game must be zero-sum, meaning that the cost for the other player (the *maximising player*) is minus the cost for the minimising player. As their names suggest, the minimising player tries to minimise the cost function, while the maximising player tries to maximise it.

When the game tree can be fully explored, the minimax strategy will find a path that guarantees the best outcome for each player when the other plays as well as possible. [5] For example, consider a game in which each player makes a single move in turn, each choosing from two possible moves. Figure 2 shows three stages of a minimax search for such a game. Stage 1 shows the value of the cost function at each leaf node. Stage 2 shows that in the left-hand, middle layer state of the game tree Player II (the minimising player) would choose the move that minimises the outcome; the game would end in the leaf state with outcome 2. Stage 3 shows Player II's decision in the right-hand state, and also that Player I would choose the move leading to the left-hand state, guaranteeing himself the largest minimal outcome.

3.2 Modelling Routing as a Game

The intuition behind modelling routing as a game is to note that the problem of routing can be understood as a contest between the network and the routers. The routers are, in effect, competing with a network that is trying to outwit

[5] When it is not feasible to explore the whole game tree, the cost function must sometimes be applied to truncated runs; in such cases it effectively judges the state of the game using a heuristic. We discuss this in Section 4.5.

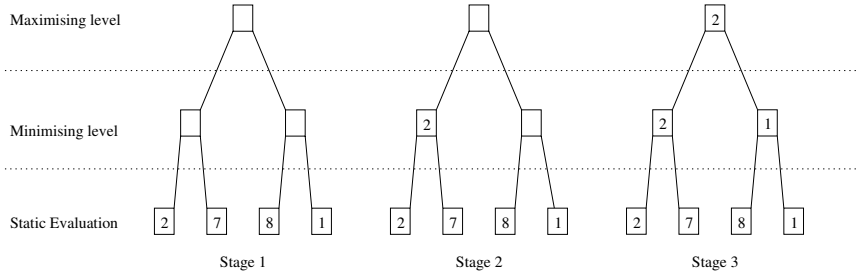

Fig. 2. The minimax algorithm at work

them. How well do the routers perform against "Murphy's network"? We are therefore considering a two-player, zero-sum game, as described above.[6] To model anything as such a game we need to:

- identify the two players and their initial states, saying which is the minimising player and which is the maximising player;
- define the game moves for each of the players;
- specify a cost function that quantifies the outcome for the minimising player; the minimising player chooses moves to minimise this function and the maximising player chooses moves to maximise it.

In all our uses of game theory the two players will be same. All the routers together form one player, which is referred to henceforth as the *set-of-routers* player; the other player is the set of links, which is called the *network* player. Game moves for the set-of-routers player are, in essence, to execute the routing protocol. And for the network player game moves are to change the network topology. This is the basic insight behind our mapping to model routing protocols as games; the sections below expand on this to describe the mappings in more detail. The cost function depends on the property being studied, and is discussed, below, for each property separately.

Once the game has been defined the game tree can be constructed and explored. The minimax strategy searches through the game tree to find the minimax path; the minimax value (or minimax outcome) is the cost function applied to this path. The meaning of the minimax value can be interpreted in the following way: within the constraints provided to the game, if the routers behave optimally, then whatever changes in the network occur, the routers are guaranteed to do no worse than the minimax value.

With these basic intuitions described we can think in a bit more detail about how game theory can be used to study the properties that interest us.

[6] This mapping is not unique; it would, for instance, be possible to construct an n-player game.

3.3 Targeting Soundness

When analysing the soundness of a routing protocol, the objective is to understand whether the routing protocol copes with a mobile network, however much it changes.

To turn this into a game we first note that the network can be understood to have won if the routers are unable to gain a correct view of the network. In our game theoretic model the game ends when the network has stopped changing and one of the two players has won. The set-of-routers wins if all routers have succeeded in gaining a correct view of the network, and the network wins if the routers are mistaken about the state of the network, and are unable to rectify this state.

The cost function of the game can be considered to be a cost predicate – "routers have a correct view of the topology in the final state of the game". The results we obtain from this analysis are of the following forms:

- the protocol copes if the predicate is eventually true; or
- the protocol does not cope if the predicate remains false, however much opportunity the routers are given to regain a correct view.

3.4 Targeting Network Overheads

The objective in studying the network overheads consumed by a routing protocol is to understand how much network traffic is required for the routing protocol to cope with a changing network. Recall from Section 2.2 that we are principally interested in comparing routing protocols, so the exact measure we use is less important than having a consistent basis for comparison. The game ends when the routers have gained a correct view of the network. It is therefore necessary for the routing protocol to be sound; i.e., routers must always be able to recover after changes in the network, in the sense described in the previous section.

The cost function of the game is a measure of the amount of traffic required by the routers to gain a correct view of the network. Unlike investigations of soundness, network traffic on each run of the game must be tracked, so the cost for a path depends on the complete path to the final state. It is then possible to produce graphs showing the network traffic required to gain a correct view, against degree of network fluctuation. In line with our objective of comparing routing protocols, the results from this analysis take the form of a comparison of network traffic graphs for different routing protocols.

3.5 Targeting Convergence

When investigating convergence of a routing protocol, the objective is to understand how quickly the routers' beliefs converge to correct (or sufficiently correct) views of the network. In this case, the game can continue indefinitely. In practice, the game is run for as long as is required for us to gain a satisfactory idea of the behaviour of the protocol.

The cost function of the game is dominated by the correctness of the routers' views of the network; the more correct the routers are, the better the routers have done. The cost function also includes a measure of the network traffic, which is used to choose between moves that lead to states with the same degree of correctness of the routers' views; the higher the network traffic is, the worse the routers have done. The network traffic part of the cost function depends on the complete path to the final state, and the (dominant) correctness part depends on only the final state.

We need some way of measuring the correctness of the routers' views of the network, and how this changes with time. We have a *correctness measure* which measures correctness for the purpose of assessing convergence. This correctness measure is calculated as an average over all routers. Graphs can be produced showing (average) degree of correctness at each state along the minimax path, against time. Again our main concern is to compare routing protocols, so the results from this analysis are a comparison of convergence graphs for different routing protocols.

4 Generic mapping issues

4.1 Overview of Mapping

The previous section outlined how games might be constructed to analyse various properties of routing protocols. In this section we look at the generic issues in defining the mapping, prior to a more detailed discussion of the specifics of the mapping for each analysis problem.

To study soundness and performance of a routing protocol we define the game as follows:

(i) the set-of-routers player, representing the *set* of all the routers, is the minimising player; the network player, representing the *set* of all the links, is the maximising player; in the initial state of the game all links are down and each router has a correct view of the network;

(ii) an atomic move for a router is to send all its routing messages, as specified by the protocol (in addition, all routers notice local link changes and process received messages). An atomic move for the network changes the

state of one link from up to down or *vice versa*. A game move, for either player, is a (small) number of atomic moves;

(iii) the cost function is a lexicographic ordering of two measures, in the following order:
  (a) the inconsistency, at the final state of the path, of the routers' views of the network topology with the actual state of the network;
  (b) the amount of network traffic used by the routers.

So, the network player aims to maximise the cost function, which records how confused the routers are about the state of the network, and how much traffic they have used; the set-of-routers player aims to minimise this same function. The lexicographic cost function described here gives the cost functions described in Sections 3.3, 3.4 and 3.5 for the three types of analysis; in particular, the inconsistency part can be ignored when the cost function is evaluated in a state of the game where the routers have correct views of the network.

We have described the basic mapping we are using, but there are a number of issues regarding the exact way this mapping is implemented; these are discussed in the next section.

4.2 Modelling Issues

As we are considering routing techniques rather than specific routing protocols, we have begun by analysing low fidelity models of real protocols. As a result we have been forced to make a number of modelling decisions regarding the mapping of routing protocols to a minimax game. We describe the most important issues here. In some cases the same decision has been taken in each of our analyses, and in some cases we have found it necessary to make different decisions according to the kind of analysis being performed.

Below we discuss the following modelling issues in some detail:

- The criteria used to decide what updates to send out.
- The representation of update data.
- Details of the distance vector routing model.
- How to deal with the *optimal scheduling* assumption.
- The exact form of the game's cost function.

One of the most significant decisions we have had to make involves the criteria a router uses to choose what updates to send out, when it is scheduled. The question is whether to model routers creating and sending out updates only in response to changes in the network, or if they should send out updates regardless of whether the network has changed (we call these two options

"output on change" and "output always" respectively). In a real routing protocol, there would most likely be a combination of the two, with updates created and sent in response to a change in the state of the network, as well as periodic broadcasting of updates. However, at this stage we have decided to use either one approach or the other, for simplicity of modelling. The particular decision made is explained in each analysis section.

Another modelling decision we have made relates to the representation of the data contained in updates. We have to choose between representing updates accurately as either link state packets (LSPs) or distance vectors,[7] and representing
them less accurately as individual link state updates or distances. There are computational efficiency reasons for using individual link state updates or distances, but it would clearly be preferable to use a more accurate representation. We explain later which option we chose in each of our analyses.

We had to make a decision relating to our distance vector routing model, which was whether distance vectors being transmitted by routers should include the entire path to a node or simply the neighbour on that route. Again different choices were made in different analyses; these are explained in the appropriate sections.

A further issue we had to address was how to model the scheduling of routers. All the routers together form the set-of-routers player, who decides which router should send routing updates in each atomic move. In effect we are assuming that the routers are under the control of a global controller that can enforce *optimal scheduling*. We use *fairness constraints* to deal with this unrealistic optimal scheduling assumption. A fairness constraint imposes a restriction on which routers are permitted to perform atomic moves at any given point in the game. When we say that we are using an $n + k$ fairness constraint, this means that each of the n routers must perform a set-of-routers atomic move at least once in every $n + k$ set-of-routers atomic moves. (We also refer to this by saying that each router must be *scheduled* at least once in every $n + k$ set-of-routers atomic moves.) We indicate in the appropriate sections where fairness constraints are being used.

Another issue is exactly what should constitute the game's cost function. The network traffic part of the function has been used consistently throughout our analyses, to represent the number of pieces of routing information sent. However, we discovered it was necessary to vary the inconsistency part of the cost function, according to the kind of analysis we were carrying out. Depending upon how difficult we wanted to make the routers' task of gaining

[7] Briefly, a link state packet records the states of all other nodes in the network; a distance vector records the distances to all other nodes in the network.

a correct view of the network, we varied the form of the inconsistency part of the function, to be somewhere between knowledge of a single link or distance, and knowledge of the entire network; the choices made are explained in each analysis section.

4.3 Detailed Description of a Set-of-Routers Atomic Move

The atomic moves and game moves for the network are (we hope) reasonably clear. The game moves for the set-of-routers are rather more complex, so we elaborate on them here in some detail. This section is not vital for understanding the approach, so the reader may choose to skip to Section 4.4.

In each atomic move one router is chosen (the choice being determined by the need to minimise the cost function that records inconsistency and network traffic). Let us say that router n is chosen, then the atomic move consists of the following sequence of activities: [8]

(i) Each router checks the state of its local links, and updates its own record of the state of those links accordingly. [9]

(ii) Each router performs some processing, as follows:
  (a) In link state routing, every router creates its own LSP and puts it on its outgoing queue (as is required by the "output always" model).
  (b) In distance vector routing, each router discards distance vectors it had received from neighbours for which the link has just gone down, and recalculates its own distance vector accordingly.

(iii) Router n performs a broadcast, which consists of the following:
  (a) In link state routing, router n processes the LSPs on its holding queue (which is empty the first time this router is scheduled). For each one, if router n is now linked to the LSP's destination router, that LSP is removed from the holding queue and added to a list of LSPs that are to be broadcast.

  Router n then processes the LSPs on its outgoing queue (which contains only router n's own LSP the first time this router is scheduled). Each LSP in the outgoing queue is cloned a number of times, with each clone corresponding to a particular destination router other than n itself and the sender of the LSP (if different from n). For each

[8] Let us assume, for the sake of simplicity, that we have chosen to represent updates as link state packets or distance vectors, and that we are using the "output always" approach.
[9] Note that in both link state routing and distance vector routing, all routers maintain a record of at least the state of their local links; in link state routing routers also maintain a view of the rest of the network, while in distance vector routing routers also maintain their own distance vector.

cloned LSP, if n is linked to the destination of that LSP, the LSP is added to the list of LSPs to be broadcast, overwriting any duplicate LSPs already there. If n is not linked to the destination of the LSP, the LSP is added to n's holding queue, as long as its timestamp is more recent than that of any LSP already in the holding queue, originating from the same source and with the same destination. Each LSP in the list to be broadcast is then delivered to its destination.

(b) In distance vector routing, router n broadcasts its own distance vector to all neighbours to which it is linked (as required by the "output always" model).

(iv) All routers that have been sent updates receive them, and act as follows:

(a) In link state routing, when a router receives an LSP it updates its view of the network accordingly. It then puts the LSP on its outgoing queue, but only if its timestamp is more recent than that of any LSP already in the outgoing queue, originating from the same source and with the same destination. (In reverse-path forwarding, the router does none of this unless it believes the sender of the LSP is on the shortest path between itself and the source of the LSP.)

(b) In distance vector routing, when a router receives a distance vector, it adds this to its list of stored distance vectors, replacing any distance vector previously received from the same neighbour. It then recalculates its own distance vector.

It is also possible for a 'null' set-of-routers atomic move to occur. In this case, no router is scheduled, and only activities 1 and 2 in the list above are performed.

4.4 Implementation

We implemented a simple Java tool that uses the above mappings to model, as a minimax game, three different routing techniques, namely link state routing [9], the reverse path forwarding (RPF) algorithm [17] for link state routing, and distance vector routing [9]. The tool allows the user to specify the number of atomic moves that comprise each game move. It then generates the game tree for that game, and searches it for the result according to the minimax algorithm.

Modelling a 7 node network, the tool can build and fully search a game tree consisting of 12 atomic moves in a few minutes. It is difficult to compare this type of bounded depth first search with the full depth first (or breadth first) search of a model-checker. However, the approach described here is yielding a degree of useful analysis on models with state spaces that are far beyond

the scope of any model-checker. The speed of the search has benefited from the use of alpha-beta pruning [18] and some simple symmetry reduction, but we have not put much effort into optimising the search tool; it is a crude and simple prototype.

4.5 Tractability of Analysis

The size of network that we are able to analyse with this technique is a good deal larger than would be possible using conventional model-checking techniques. However, the exponential growth of the game tree is a seriously limiting factor on how deep in the tree it is possible to search in a reasonable period of time.

We have addressed this problem by implementing an additional facility in our game search tool. This facility allows us to perform recursive runs of a game, where the initial state of any run (except the first) is the final state of the previous run. This is not equivalent to a complete search of the game tree; rather, it provides an estimate of the state of the game, using a heuristic. This technique enables us to search arbitrarily deep in the game tree.

5 Analysing soundness

The first routing protocol characteristic we analysed was soundness. A routing protocol is certainly not sound if the routers can be 'beaten' by the network. However, the converse does not hold. A failure of the network to win does not constitute a verification of the protocol, even merely for the size of network modelled; this contrasts with model-checking.

To see this, remember that the set-of-routers player acts as a global controller, deciding which router should send routing updates in each atomic move – this is the optimal scheduling assumption mentioned earlier. An important consequence of this assumption is that our technique is currently a refutation procedure: if the set-of-routers can force a win then the routing protocol may still be flawed, since a non-optimal scheduling order might cause the routers to fail; on the other hand, if our technique finds that the set-of-routers must lose (even under the optimal scheduling assumption) then there is a genuine problem for the routing protocol.

5.1 How We Used the Tool

When we were considering how to use our tool to analyse the soundness of a routing protocol, the question that had to be asked was: what would actually constitute a victory for the network, or in other words, how would an

unsoundness manifest itself in the output of the tool?

The answers lay in the inconsistency part of the cost function. A non-zero inconsistency value indicates that at least one router has, at that point in the game, been unable to gain a correct view of the network. It is to be expected that routers will become confused about the state of the network at times during the game. However, if a routing protocol is to be considered sound, one would expect that if, at any time, the network were to be fixed at any particular state, each router should eventually converge to a correct view of at least the partition of the network in which it is located.

We observed therefore that if, with routers behaving according to some specified routing protocol, it is possible to find a state of the game in which at least one router is unable to regain a correct view of the (no longer changing) network by the end of the game, then that protocol can be considered unsound. This is what we consider constitutes a victory for the network. [10]

We carried out a number of experiments with two routing protocols, namely link state RPF and distance vector routing. We provided the tool with various numbers and combinations of network and set-of-routers atomic moves, trying to find victories for the network, i.e. situations where routers become irretrievably confused. Our results are detailed below.

5.2 Decisions

The inconsistency part of the cost function is:

- for link state routing, correctness of the believed state of the link between node 0 and node 1;
- for distance vector routing, correctness of the believed distances to node 0 and to node 1.

The messages take the form of individual updates and distances, rather than packets and vectors, and we use the "output on change" model. We do not use fairness constraints.

This information is summarized in Table 1.

5.3 Results

We began by considering the link state RPF algorithm. However, the tool could not find any flaws in this algorithm, so to validate our approach we tweaked the models so that:

- the algorithm was wrong, or

[10] We used a bounded depth search, which is discussed in Section 8.

- individual routers could behave incorrectly.

In both cases the game search found ways in which the network could beat the set-of-routers.

We then investigated distance vector routing. It is well known in the routing community that distance vector routing in its most simple form suffers from the Count to Infinity Problem [9], whereby routers can forward packets to each other infinitely often because of mistaken local beliefs about the state of the network. (For example, router A might believe the shortest route to C is via next hop B, while B believes the best route to C is via A.)

In this case, the tool reported that the network could force a win, so we were able to conclude that such confusion could eventually arise. Inspection of the state reached on the path to the minimax solution revealed that counting to infinity was possible, which we had not expected for the particular initial state and operating assumptions modelled. Similarly, the tool found an example of counting to infinity occurring even for the Split Horizon [9] variant of distance vector routing.

These experiments served to validate our approach; they provided evidence that the tool is capable of finding surprising behaviours.

6 Analysing network overheads

We felt that the work on game search to study soundness was in itself novel and interesting, but our interest in minimax search as a basis for understanding routing increased greatly when we realised how to use game search to study network overheads.

6.1 How We Used the Tool

The simple observation is that the minimax value calculated of a game between the routers and the network can serve as a potentially effective characterisation of the routing protocol's performance. Recall that we are using a cost function that records the number of routing update packets that have been sent by routers. If we work out how many update packets are required for the routers to achieve a correct view of the network, then we have a measure of performance. This cost function treats each routing update packet as having the same cost, and it assumes a uniform link quality. Even so, it does provide a basic measure of how much network resource is consumed to support routing.

Notice that we now need to evaluate the cost function at states of the game along the whole path, not merely at the final state of the path. This distin-

guishes it from the cost function we used when mapping soundness analysis to a game.

It is widely believed that RPF consumes lower network overheads than flooding (indeed inspection of the algorithms indicates that this is likely to be the case). Nevertheless, we have not been able to find in the literature any *analytical* quantification of the savings achieved by RPF. In line with our aim of rigorously analysing routing techniques, we have used game theoretic analysis to quantify the savings achieved by RPF. We have also used these techniques to investigate the network overheads required by distance vector routing.

Each game encodes a particular initial state and constraints on the number of consecutive atomic moves allowed within each player move. Thus all questions asked of the tool are with respect to a particular initial state and particular move constraints.

The tool was used to determine how many set-of-routers atomic moves are needed to cope with the network changes (i.e., to achieve a correct belief at each connected node about the state of a particular link) for various numbers of alternating network and set-of-routers moves. The cumulative amount of routing traffic required was measured at regular intervals throughout the game.

6.2 Decisions

The choice of the inconsistency part of the cost function was affected by computational efficiency considerations. If all the routers were expected to gain a correct view of the entire network after every set-of-routers move, then it would be necessary to search very deep in the tree, which would become computationally intractable. Therefore we set the inconsistency part of the cost function to the less stringent choice that was used for the soundness analysis; it is reproduced below:

- for link state routing, correctness of the believed state of the link between node 0 and node 1;
- for distance vector routing, correctness of the believed distances to node 0 and to node 1.

The messages take the form of individual updates and distances, rather than packets and vectors, and we use the "output on change" model. We have produced results both with and without the use of fairness constraints. [11]

This information is summarized in Table 1.

[11] We also constructed a model using the "output always" approach, and where the messages were LSPs and distance vectors; however, the results obtained from that model were less conclusive than those presented here.

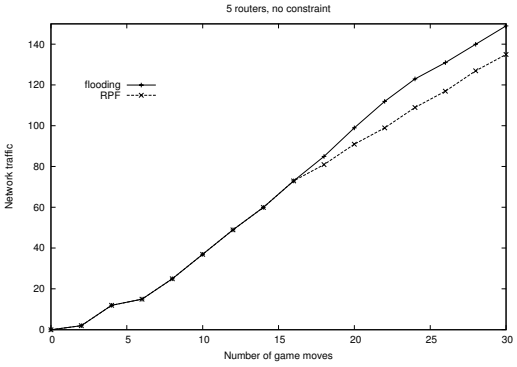

Fig. 3. Comparison of traffic for RPF and flooding

6.3 Results

Figure 3 shows how the total routing traffic communicated to regain the correct network view varies with increasing numbers of moves.

A reduction in network overheads consumed by RPF compared to flooding is indeed apparent in Figure 3, but we were surprised by the small improvement shown; we had expected a larger gap between the two graphs. On reflection we formed the opinion that the small gap size was likely to be an artefact of our mapping to a game; we hypothesised that the blame lay with the (unrealistic) full control the set-of-routers player has to schedule individual routers. The intuition is that realistic results are more likely to be exhibited if some unrealistic power is removed from the set-of-routers player, in this case the *unconstrained* power to choose which router to schedule for each atomic move. (Indeed, it is credible that a set-of-routers player with full control of when to schedule routers can take better advantage of this power if routers flood messages than if they use RPF: the set-of-routers player can choose to schedule them in the same order as they are optimally scheduled when running RPF, thus minimising the non-RPF communications.)

In reality, of course, routers operate concurrently and often on similar processors under similar load, so they may be considered to be scheduled at similar rates. This suggested to us that more realistic results would emerge if the set-of-routers were constrained to be scheduled in a fair way. So, as mentioned in Section 4, we opted to formulate fairness constraints that require each router to be scheduled at least once in any sequence of $n + k$ successive set-of-routers atomic moves, where n is the number of routers, and k is a number that characterises the degree of fairness.

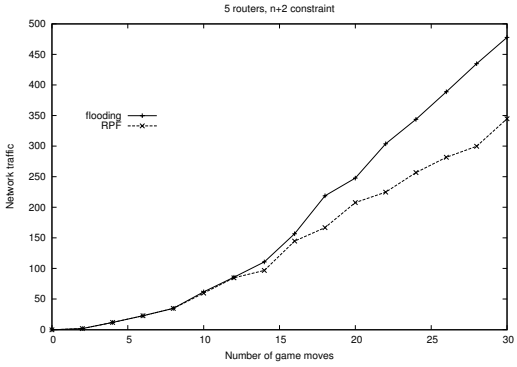

Fig. 4. Comparison of traffic for RPF and flooding with fairness constraint

We tested our hypothesis by extending the tool to allow such fairness constraints to be enforced. We then generated new pairs of graphs, with varying degrees of fairness enforced. Figure 4 shows the graphs for the case where 5 routers must each be scheduled in any sequence of 7 set-of-routers atomic moves. As hypothesised, the results show a larger reduction in the network overheads consumed by RPF over flooding than when there are no fairness constraints.

We concluded our investigation of network overheads by measuring the overheads consumed by distance vector routing. It is difficult to directly compare results obtained for distance vector routing with our prior results for link state routing, as the inconsistency parts of the respective cost functions are, by necessity, different. Nevertheless, our results for distance vector routing, with and without fairness constraints, are displayed in Figure 5.

7 Analysing convergence

Having had some success in our analysis of soundness and network overheads, our attention then turned to the one remaining routing protocol property mentioned in Section 2.2, namely convergence.

7.1 How We Used the Tool

It is widely believed that link state routing converges more quickly than distance vector routing, and that link state flooding converges more quickly than link state RPF[9]. However, again we have been unable to find any analytical quantification of this. We felt that game theory could be usefully applied

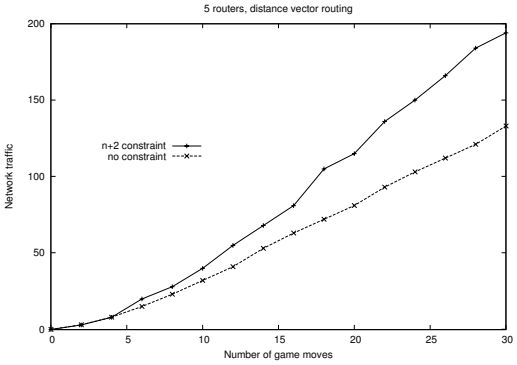

Fig. 5. Traffic required by distance vector routing, with and without fairness constraint

	Soundness	**Network Overheads**	**Convergence**
Message format	individual update / individual distance	individual update / individual distance	full link state packet / full distance vector
Output model	output on change	output on change	output always
Form of cost function	cost predicate	cost function	cost function
Correctness part of cost function	link between 0 and 1 / distances to 0 and 1	link between 0 and 1 / distances to 0 and 1	whole network / whole distance vector
Evaluation of cost function	at final state	along whole path	along whole path
Fairness constraint	without	with and without	with and without
Distance vector model	not entire path	report entire path	report entire path
Correctness metric	n/a	n/a	% links correct / % neighbours correct

Table 1
Summary of modelling decisions (link state routing / distance vector routing)

to address these questions, and indeed we have been able to use game theoretic techniques to carry out an analytical comparison of the convergence of a number of different routing protocols.

However, it was not immediately obvious how we could use our game search tool to quantify the convergence of a routing protocol. It was clear that the network convergence cost function did not provide us with enough information to do this, and that we needed our tool to provide us with more detailed output of the routers' state throughout the game. We therefore enhanced the functionality of the tool, so that it calculates how accurately each router views the network at each time index during the game, according to some correctness metric. The precise metric used varies according to the routing protocol being analysed, because the information maintained at routers depends on the particular routing protocol.

In constructing a game to analyse convergence, we decided to use alternate network and set-of-routers moves, where the number of atomic moves for each was fixed up-front. The number of set-of-routers atomic moves was carefully chosen to give the routers reasonable opportunity to recover a "good enough" view of the network before the network changed state again. In this case we decided that "good enough" did not mean that all routers had to regain a completely correct view.

7.2 Decisions

Unlike our analysis of network overheads, here we do not require all routers to gain a correct view of the network, so the computational efficiency of the search is not an issue. Therefore the following choices were made for the inconsistency part of the cost function:

- for link state routing, does each router have a completely correct view of the network?
- for distance vector routing, does each router have a completely correct distance vector?

The correctness metric is defined as follows:

- for link state routing, the average percentage of links about whose state routers have a correct belief.
- for distance vector routing, the average proportion of neighbours in the routers' distance vectors that are correct.

The messages take the form of LSPs and distance vectors, and we use the "output always" model. We have produced results both with and without the use of fairness constraints.

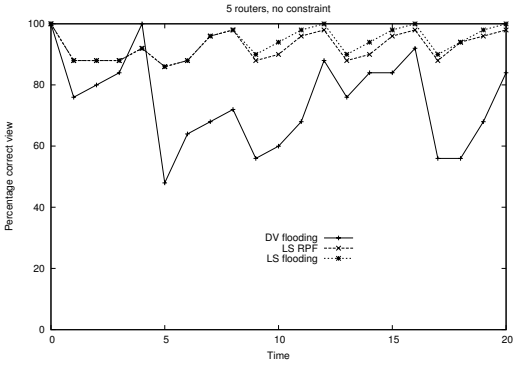

Fig. 6. Comparison of convergence of distance vector routing, link state RPF and link state flooding

This information is summarized in Table 1.

7.3 Results

Figure 6 shows how the routers' view of the network varies during a game in which the network and set-of-routers are alternately given two atomic moves and four atomic moves respectively. To understand what the graph is showing us, it is important to realise that the time index is incremented after every set-of-routers atomic move, so the network changes state between time 4 and time 5, between time 8 and time 9, and so on.

We see that after a network move routers become confused and then recover some way towards a correct view of the network; comparing graphs, it can be seen that the degree of confusion and the rate of recovery are somewhat worse in distance vector routing than in link state RPF. The same is true, but to a lesser extent, for link state RPF and link state flooding respectively. It is difficult to give a precise quantification of convergence from Figure 6, but it is at least possible to gain some idea of the relative speeds of convergence of the different routing protocols.

Figure 6 was obtained without the use of a fairness constraint, which, as has been observed before, could mean that the set-of-routers player were allowed unrealistic power to schedule particular routers. To solve this problem we re-ran the game as before, except that this time we imposed an $n + 2$ fairness constraint (so that all 5 routers must be scheduled in any sequence of 7 set-of-routers atomic moves). The results from this are shown in Figure 7.

This graph is broadly similar; the main observation to be made is that there is a larger gap between link state RPF and link state flooding. We believe

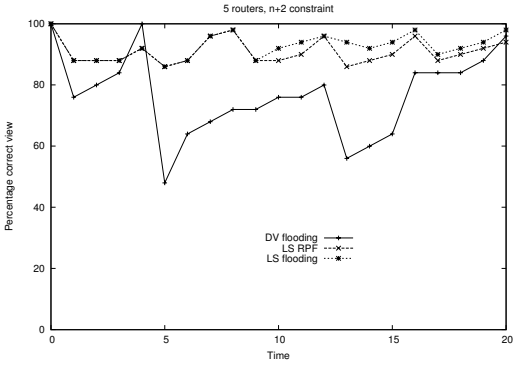

Fig. 7. Comparison of convergence of distance vector routing, link state RPF and link state flooding with fairness constraint

this is a more realistic reflection of the relative speeds of convergence of the two protocols. Our analysis has confirmed prior beliefs about convergence of routing protocols, which we again feel validates the game theoretic approach.

8 Limitations and further work

There are a number of limitations of this approach:

(i) *Perfect information.* The minimax game is a game of perfect information, as in chess. In simple terms this means that each player is assumed to have full knowledge of the state of the game. This is clearly a strong assumption. Our current research is investigating mapping to games of *imperfect information.*

(ii) *Optimal scheduling of routers.* The assumption that routers are optimally scheduled is certainly a limitation of the approach; it makes it a refutation procedure. However, we believe this can be addressed within the current framework by imposing fairness constraints, or considering games of chance.

(iii) *Finite number of nodes.* Our ability to model only about 7 or 8 routers is much better than we would expect with a model-checker, but it is still a far cry from the thousands of nodes found in a real network. There are various approaches to this problem, which will be reported in a future paper.

(iv) *Bounded depth.* It has been argued that a major limitation of this approach is that the depth of the game tree is bounded. We may have a

'result' for our bounded depth, but that gives no guarantee against the possibility of the model exhibiting very different behaviour one ply deeper in the game tree. We plan to address this problem by demonstrating that the cost function is continuous with respect to the depth bound.

(v) *Determinism.* Our models have no notion of probabilities associated with game moves, which is a further limitation. This relates to games of chance.

(vi) *Details of mapping.* Several choices arise when deciding how to map a particular routing technique to a game. There is an obligation to ensure that these choices are appropriate for the property investigated. More work is planned to determine the most suitable choices.

Further work will combine a study of these limitations and ways to address them along with continued modelling of routing techniques and game theoretic quantification of their performance properties.

9 Conclusions

This paper has argued that rigorous analysis of ad-hoc routing protocols can be achieved by mapping the protocols to simple games and calculating their minimax outcomes. We have illustrated the approach by analysing key correctness and performance characteristics of some basic routing techniques. Our main motivation has been to understand the bounds on performance of routing protocols. We feel that game theory has all the conceptual tools necessary for the rigorous analysis of ad-hoc routing protocols. In addition, game theory has the advantage of scaling rather better than might be expected with model-checkers.

10 Acknowledgments

We thank Phil Armstrong, Christie Bolton, Sadie Creese, Michael Goldsmith, Gavin Lowe, Colin O'Halloran, Bill Roscoe, Einar Vollset and Paul Whittaker for interesting discussions.

References

[1] Mobile Ad-Hoc Networks Charter.
http://www.ietf.org/html.charters/manet-charter.html.

[2] OPNET Modeler.
http://www.opnet.com/products/modeler/home.html.

[3] The network simulator – NS-2. http://www.isi.edu/nsnam/ns.

[4] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia and M. Gerla. GloMoSim: A Scalable Network Simulation Environment. Technical Report 990027, UCLA Computer Science Department, May 1999.

[5] David Cavin, Yoav Sasson and André Schiper. On the Accuracy of MANET Simulators. Proceedings of the Workshop on Principles of Mobile Computing (POMC'02).

[6] Kwan-Wu Chin, J. Judge, A. Williams and R. Kermode. Implementation Experience with MANET Routing Protocols. ACM SIGCOMM Computer Communications Review, Volume 32, Number 5: November 2002.

[7] Lundgren, H., E. Nordstrom, C. Tschudin. Coping with Communication Grey Zones in 802.11b based Ad-hoc Networks. Proceeding of ACM WoWMoM'02, September 2002, Atlanta, Georgia.

[8] T. Camp, J. Boleng and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, September 2002.

[9] R. Perlman. Interconnections Second Edition: Bridges, Routers, Switches and Internetworking Protocols. Addison-Wesley, 1999.

[10] A. W. Roscoe. The Theory and Practice of Concurrency. Prentice-Hall International, 1998.

[11] E. M. Clarke, O. Grumberg and D. Peled. Model Checking. The MIT Press, 1999.

[12] Bowen Alpern and Fred Schneider. Defining Liveness. Information Processing Letters, 21(4):181-185, October 1985.

[13] L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker and R. Segala. Symbolic Model-checking of Concurrent Probabilistic Systems using MTBDDs and the Kronecker Representation. Proceedings of TACAS 2000, LNCS 1785, pp 395-410. Springer, 2000.

[14] J. von Neumann and O. Morgenstern. The Theory of Games and Economic Behavior. Princeton Univ. Press, Princeton NJ (2nd ed., 1947).

[15] Drew Fudenberg and Jean Tirole. Game Theory. The MIT Press. October 1991.

[16] Osborne, Martin J. An Introduction to Game Theory. Oxford University Press Inc, USA. Januaury 2004.

[17] S. Deering. Multicast Routing in a Datagram Internetwork. PhD Thesis. Stanford University, California, 1991.

[18] D. E. Knuth and E. W. Moore. An Analysis of Alpha-Beta Pruning. Selected Papers on the Analysis of Algorithms. CSLI Press and Cambridge University Press, November 2000.