Towards an Algebra of Routing Tables

Peter Höfner[1,3] Annabelle McIver[2,3]

[1] Institut für Informatik, Universität Augsburg, Germany
[2] Department of Computing, Macquarie University, Australia
[3] National ICT Australia Ltd. (NICTA)
peter.hoefner@nicta.com.au, annabelle.mciver@mq.edu.au

Abstract. We use well-known algebraic concepts like semirings and matrices to model and argue about Wireless Mesh Networks. These networks are used in a wide range of application areas, including public safety and transportation. Formal reasoning therefore seems to be necessary to guarantee safety and security. In this paper, we model a simplified algebraic version of the AODV protocol and provide some basic properties. For example we show that each node knows a route to the originator of a message (if there is one).

1 Introduction

Wireless Mesh Networks (WMNs) are currently used in a wide range of application areas, including public safety, transportation, mining, etc. Typically, these networks do not have a central component (router), but each node in the network acts as an independent router, regardless of whether it is connected to another network or not. They allow reconfiguration around broken or blocked paths by hopping from node to node until the destination is reached.

The Ad-hoc On-Demand Distance Vector Routing (AODV) protocol [20] is a routing protocol that finds a route on demand (if needed and if unknown) in WMNs. To guarantee safety and security aspects for WMNs in general and AODV in particular, mathematical analysis should be used. It is our belief that one does not have to develop new theories. We believe that the concepts and tools developed over the last decades are powerful enough to capture new concepts like AODV. For example, Singh et al [21] use process algebra to model a simpler version of AODV. Within this approach processes reflect the behaviour of nodes, i.e., a process describes how nodes react when messages are received. In this paper we will use routing tables as basic elements rather than processes. This allows us to use algebraic concepts like semirings, Kleene algebra and matrices to model some parts of the AODV protocol. The algebraic operations will then be transformers of routing tables. We show how the basic concepts of AODV can be modelled. In particular, we characterise AODV control messages, which are used to distribute knowledge through the network. To achieve this goal, the algebraic operations have to model things like broadcasting (sending to all neighbours) and unicasting (sending to one neighbour) messages.

2 AODV and its Design Challenges

The AODV protocol [20] supports routing between nodes in a WMN in a demand driven manner. Nodes are able to communicate directly between connected neighbours, but routes between nodes in general comprise a number of "hops". In a dynamic environment, where the network topology can change—typically a consequence of mobile nodes which can cause new connections to be established, or existing ones to disappear—a routing protocol must be able to identify routes which have become obsolete and replace them with valid alternatives.

In order to establish valid routes, information is disseminated throughout the network by the exchange of control messages (requests, replies and error messages) and over time a node essentially "learns" to which distant nodes it is connected, albeit indirectly by multiple hops. One of the challenges presented by a dynamic topology is the management of routes if they become obsolete. Without a careful accounting policy to handle "broken" routes—even if the specific link failure is someway downstream—there is a risk that the routing algorithm establishes so-called "loops": routes which, if followed, will never reach their destination. Reporting broken links to other nodes on a route is one of the functions of AODV and the source of much of its complexity.

In this paper our aim is to describe an algebraic approach to model the behaviour of the network control messages, with a view to modelling some of the principal features of AODV. We begin in the next section by an overview of the underlying network mechanisms and how they fit together.

2.1 Overview of AODV

The basic algorithm underlying AODV disseminates knowledge of local connections or "links" throughout the network; nodes "know" their immediate neighbours (to whom they are connected directly), or "1-hop neighbours" via a network "polling" mechanism known as the *Hello protocol.* Its task is to broadcast a "hello" regularly to any node in its vicinity; any node receiving the broadcast establishes a 1-hop connection to the sender. Any node D which is not directly connected to a source node S requires messages destined for D to be routed via a series of intermediate nodes. We say a *route* from S to D is a sequence of nodes

$$S, N_1, N_2, ..., N_k, D \ , \qquad (1)$$

where each successive node in the sequence is a 1-hop neighbour to its predecessor. The route of Expression (1) has $k{+}1$ hops. Such a route is said to be *loop-free* when no node occurs more than once.

The objective of AODV is to ensure that, when required, a node is able to identify its next immediate hop on a complete valid route to a given destination D. Nodes maintain a *routing table entry* for each possible destination, which includes the next hop together with the total number of hops required to reach the destination. For example the routing table at node S would have its next hop for destination D registered as N_1 with a hop count $k{+}1$, whereas each intermediate node N_i ($1 \leq i < k$) would have its next hop to D registered as N_{i+1} with a hop count $k-i+1$. As we shall see the hop count is necessary for the node to choose between two or more possible valid routes.

Establishing the routing table entries is implemented by flooding information about the 1-hop neighbours throughout the network; it has been shown that the next hop and the total hop count is not sufficient information to avoid the accidental construction of loops. We sketch how that might occur.

Information about the topology, i.e., the 1-hop neighbours, is flooded throughout the network by exchange of control messages, allowing knowledge of routes to be established and next hops to be recorded in the respective routing tables.

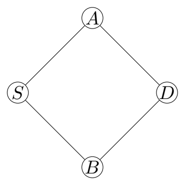


Fig. 1. Illustrating changing topology

Consider the scenario in Figure 1. Here there are two valid routes from S to D (via A and B, resp.). Following the specification of the protocol each node is able to store exactly one path to a certain destination. Assume that S has established a path S, A, D. If the link between A and D subsequently breaks, and A initiates a new search for a path to D, its request for a new path would be answered by a response from S that a route can be found to D via A. If fresh requests are not distinguished from previous requests, this would establish a path which is not loop free. To avoid this scenario AODV uses a scheme to distinguish between fresh requests and old routes; in this paper we will concentrate on setting up the algebra to deal with the simpler case, where freshness is not considered. However, we will discuss extensions which will allow us to deal with the full generality.

2.2 Basic protocol

As for most routing protocols, the basic idea of AODV is inspired by traditional path search such as Dijkstra's shortest path algorithm [7]—AODV aims to minimise hop count. Recall that Dijkstra's shortest path algorithm keeps a record of the paths from a source S, extending those paths in each iteration of the algorithm. Loops are avoided by marking nodes that have been visited before. AODV is based on a distributed version of this idea, where the set of visited nodes is extended in several steps by utilising the network control messages. We observe that messages are either *broadcast* or *unicast*, with the difference being that broadcast messages are intended for any node that is connected, whereas unicast messages are addressed to a single specific node. Thus unicast messages are only received by the addressee, and any other node which picks up the message simply ignores it.

The main AODV control messages are as follows:

Route Request (RREQ) When a node requires a route to a (new) destination D, the node broadcasts a RREQ for that destination. A node A that receives the RREQ *either* broadcasts the RREQ again (thereby informing a possibly new set of neighbours) *or* it responds with the "route reply" (see below). The choice to broadcast is made in the case that A itself does not know a route to D; it sends a route reply if it already knows a route to D. Finally if A has already received and broadcasted or responded to a RREQ to D it simply ignores it. [1]

Route Reply (RREP) If R is the destination D or knows a route to D then A unicasts a RREP back to the originator of the corresponding RREQ, with the information about the number of hops to D.

Hello messages are used to establish 1-hop neighbours. A node executing a Hello simply broadcasts a message; any node that picks it up establishes a 1-hop connection.

Route Error (RERR) As mentioned above, WMNs are based on wireless networks with changing topology. Therefore links can be lost. When a link break in an active route is detected, a RERR message is used to notify other nodes that the loss of that link has occurred.[2]

3 Algebraic Approaches for Routing Protocols

We are not the first who want to use algebraic techniques to describe routing procedures. Most of the purely algebraic approaches are done by Sobrinho and Griffin [22, 23, 10, 9]. It seems that Sobrinho was the first who brought algebraic reasoning into the realm of hop-by-hop routing [22]. He uses algebraic properties to argue about the relationship between routing algorithms and Dijkstra's shortest path algorithm. The author states that he follows an algorithmic approach while others use a matrix approach [4]. On the other hand, Griffin uses algebraic structures like semigroups and semirings for the analysis of path vector protocols like the Border Gate Protocol BGP.

In fact, the above mentioned matrix approach is older still and was already used by Backhouse and others [3]. The next evolutionary step of the matrix approach was taken by Kozen [14] who showed that $n \times n$-matrices over Kleene algebras again form a Kleene algebra. He was able to show the correctness of a simple algorithm for reflexive transitive closure purely algebraically, in addition to establishing the relationship between matrix algebra and algorithms for all-pair shortest path algorithms. Later, Möller used test elements—now well-established in semirings and Kleene algebra—to model nodes within the matrix

[1] The RREQ handling is more complicated in the case of dealing with broken links.
[2] We include this for completeness.

algebra and to present algebraic versions of the algorithms of Dijkstra and Floyd-Warshall [18]. Another area where matrix algebras can be applied is in static program analysis. This was shown by Fernandes and Desharnais [8].

In this paper, we will extend the matrix approach aiming at a model for the distance vector protocol AODV. In particular, matrices are used as representatives of (local) routing tables, and we formulate AODV control messages as operations on routing tables. To the best of our knowledge, both Sobrinho and Griffin consider static routing protocols, i.e. where routers are required. In this setting link breaks are possible but changes in topology are not, unlike the context for AODV where it is common for links to break and the underlying network topology to change. Moreover, we are interested in modelling concrete actions performed by AODV in an algebraic setting. Again to the best of our knowledge this has not been done before. Interestingly, though path and distance vector protocols are different in practice, the underlying algebraic structures bear strong similarities.

4 Semirings, Kleene Algebras and Extensions

In this section we briefly recapitulate the algebraic structures needed to model AODV control messages.

It is well known that semirings and Kleene algebras model sequential composition, (non-deterministic) choice and finite iteration. They provide the appropriate level of abstraction for modelling actions, programs or state transitions under non-deterministic choice and sequential composition in a first-order equational calculus. A first-order calculus allows the use of first-order automated theorem provers to validate and verify properties. It has been shown that the algebraic structures used are particularly amenable to automated reasoning [13].

An *idempotent semiring (i-semiring)* is a quintuple $(S, +, \cdot, 0, 1)$ such that $(S, +, 0)$ is an idempotent commutative monoid, $(S, \cdot, 1)$ is a monoid, multiplication distributes over addition from the left and right and 0 is a left and right zero of multiplication. The *natural order* $\leq$ on S is given by $a \leq b \Leftrightarrow_{df} a + b = b$. It induces an upper semilattice in which $a + b$ is the supremum of a and b, and 0 is the least element.

A *Kleene algebra* is an idempotent semiring S extended by an operation $^* : S \to S$ for iterating an element an arbitrary but finite number of times. Such an operation has to satisfy the *star unfold* and the *star induction* axioms

$$\begin{aligned} 1 + a \cdot a^* = a^* \ , &\qquad b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c \ , \\ 1 + a^* \cdot a = a^* \ , &\qquad b + c \cdot a \leq c \Rightarrow b \cdot a^* \leq c \ . \end{aligned}$$

If elements of semirings/Kleene algebras characterise single transitions, whole transition systems are usually encoded in matrices. To calculate with matrices we recapitulate a well known result:

Theorem 4.1 *Standard operations of matrix addition and multiplication turns the family $M(n, K)$ of $n \times n$ matrices over a Kleene algebra K into a Kleene* *algebra again; the zero matrix is neutral w.r.t. $+$, and the identity matrix I w.r.t. multiplication.*

A proof can e.g. be found in [15], the techniques used are based on preliminary work by Conway [5] and Backhouse [2].

Tests of a program or sets of states of a transition system can also be modelled in this setting. A *test* in an i-semiring S is an element of a Boolean subalgebra $\mathsf{test}(S) \subseteq S$ such that $\mathsf{test}(S)$ is bounded by 0 and 1 and multiplication $\cdot$ coincides with lattice meet. We will write $a, b, \ldots$ for arbitrary semiring elements and $p, q, \ldots$ for tests. Moreover, $\neg p$ denotes the complement of p in the Boolean subalgebra. An important property (e.g. [6]) of tests is

$$p \cdot a \cdot q \leq 0 \Leftrightarrow a \cdot q \leq \neg p \cdot a \ . \tag{2}$$

In the matrix model, the maximal test set contains all diagonal matrices with tests of the underlying algebra on its diagonal. Hence Theorem 4.1 also holds for Kleene algebras with tests.

Idempotent semirings admit at least the test algebra $\{0, 1\}$ and can have different test algebras. On test semirings one can define a generalised notion of the weakest liberal precondition wlp (see [19]). This is achieved by defining a box-operator $|\_] : S \mapsto (\mathsf{test}(S) \mapsto \mathsf{test}(S))$ by

$$p \leq |a]q \Leftrightarrow p \cdot a \cdot \neg q \leq 0 \ , \qquad \text{and} \qquad |a \cdot b]p = |a](|b]p) \ .$$

The diamond $|\_\rangle$ is the de Morgan dual of this operation, i.e., $|a\rangle p = \neg|a]\neg p$. This operator will be used to determine whether there is a route from p to q in a (checking $p \leq |a\rangle q$). An important property of $|a\rangle p$ is that it is the least left preserver of $a \cdot p$. In particular, we have

$$|a\rangle p \cdot a \cdot p = a \cdot p \ . \tag{3}$$

Note that the diamond- and the box-operator bind stronger than addition and multiplication. Test semirings equipped with $|\_]$ and $|\_\rangle$ are called *modal*. Modal semirings can easily be extended to modal Kleene algebras without any further assumptions.

5 Routes and Routing Tables

A routing table is a data structure (often in the form of a table or vector) that lists the routes to network destinations and, most often, also additional information like metrics, sequence numbers or knowledge about the topology.

In classical networks/protocols a routing table is stored in a central component like a router. Reactive network protocols like AODV use a different approach: Each component has its own routing table. This avoids the existence of a central component and is therefore less prone to computer failures like crashes: If a single component crashes the remaining components can still communicate.

In AODV, each component of an arbitrary routing table stores for a known destination, the next node (where to send the packet), the length (hop count) of the route, information about the freshness of a route and some more information. For the moment we only focus on the former two components and omit the remaining (cf. Sections 2 and 7):

– The *next hop* identifies a neighbour where the packet has to be sent to reach a particular destination D.
– The *hop count* specifies the length of the path to D (i.e., number of nodes that have to be visited)

The set of node names is denoted by N; the set H gives a value to compare the quality of two entries (for example the length).

Formally, a *routing table entry* (*entry* for short) is a pair (m, x) over two totally ordered sets $(N, \preceq)$ and $(H, \sqsubseteq)$. On the set $N \times H$ of all entries, we will use the lexicographical ordering (again a total order) to measure the quality of entries.

Next to these "proper" elements, we define two special entries $(\varepsilon, 0)$ and (ε, ∞)[3], both not elements of $N \times H$. The set $\mathbb{M} =_{df} N \times H \cup \{(\varepsilon, \infty), (\varepsilon, 0)\}$ denotes the set of all possible route entries.

In AODV, the names are given as IP-addresses; its ordering is straight forward. A proper entry $(m, x) \in N \times H$ states that there is a path (its source and destination will be encoded somewhere else) with next node m and length x; $(\varepsilon, 0)$ states that there is a trivial path of length 0; hence no next hop must be given. In contrast to that, (ε, ∞) denotes an entry that corresponds to a route of infinite length, where also no next hop is given. This construct can be seen as the statement of "no (known) route".

Example 5.1 For the upcoming examples we use $N =_{df} \{A, B, C, \dots\}$ together with the lexicographical order. Moreover, we set $H = \mathbb{N} - \{0\}$. Assume $A \in N$ to be a name, then the pair $(A, 5)$ means that there is a route (its source S and its destination D will be encoded somewhere else) where the next node on the path is A and the length is 5 (cf. Figure 2). $\square$

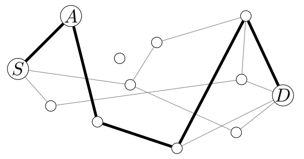


Fig. 2. A route from S to D of length 5

[3] We only use pairs to be consistent with proper elements (m, x). The symbols $(\varepsilon, 0)$ and (ε, ∞) just denote new elements and could be denoted differently.

Next we want to define operations on entries. If there is a choice between two routes going from the same source S to the same destination D, it is obvious that one should prefer the shorter route; if the lengths of two entries are the same, we take the one with the "better name" (using the ordering of names). Again, this is expressed by defining choice (addition) via the lexicographical order[4]:

$$(m,x)+(n,y) =_{df} \begin{cases} (m,x) \text{ if } (x \sqsubset y) \vee (x = y \wedge m \preceq n) \\ (n,y) \ \text{ otherwise} \ , \end{cases}$$

for $(m,x),(n,y) \in N \times H$ being proper entries, where $x \sqsubset y \Leftrightarrow_{df} x \neq y \wedge x \sqsubseteq y$. For the previously defined special elements $(\varepsilon,0)$ and (ε,∞) choice is defined by

$$\begin{aligned} (\varepsilon,\infty)+r &=_{df} r+(\varepsilon,\infty) =_{df} r \ , \\ (\varepsilon,0)+r &=_{df} r+(\varepsilon,0) =_{df} (\varepsilon,0) \ , \end{aligned}$$

for all $r \in \mathbb{M}$. Intuitively, this means that $(\varepsilon,0)$ is the best entry; an entry of length 0 is favoured over all other entries (and routes). In contrast to that (ε,∞) is the worst.

Using pairs together with lexicographical ordering in the setting of internet routing is not new; Sobrinho used similar pairs which yield "most-reliable-shortest paths" [22]. However, he does not use the concept of next hops. This concept is mentioned in [10].

Next to choice we also have to combine routes.

Example 5.2 Assume an entry $(A,5)$ from S to S' and another entry $(B,7)$ from S' to D, then there is a "route" $(A,12)$ from S to D with next hop A. A sketch is given in Figure 3. $\square$

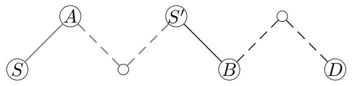


Fig. 3. Route composition

Formally, we assume N, H not only to be ordered sets, but also to offer binary operations $\circ : N \times N \to N$ and $\bullet : H \times H \to H$. Entry composition is then defined by pointwise lifting. For $(m,x),(n,y) \in N \times H$ that means

$$(m,x)\cdot(n,y) =_{df} (m \circ n, x \bullet y) \ .$$

The special elements are handled, for all $r \in \mathbb{M}$, by

$$(\varepsilon,\infty)\cdot r =_{df} r\cdot(\varepsilon,\infty) =_{df} (\varepsilon,\infty) \ , \qquad (\varepsilon,0)\cdot r =_{df} r\cdot(\varepsilon,0) =_{df} r \ .$$

[4] Note that we define the lexicographical order in a right-to-left way. That means that we first compare the second component, whenever the second component is equal, we have a closer look at the first one.

In the examples, the operation $\circ$ coincides with "taking the left element", i.e., $m \circ n = m$ for all $m, n \in N$ and $\bullet$ with ordinary addition on natural numbers.

So far, we have defined a number of useful operations on routing table entries. Under certain circumstances the above operations behave well and form a semiring and a Kleene algebra; in general we get:

Theorem 5.3 *Assume two totally ordered sets $(N, \preceq)$ and $(H, \sqsubseteq)$ with an isotone binary operation $\circ$ on N and a binary operation $\bullet : H \times H \to H$ that is* strictly *isotone, i.e., $a \sqsubset b \Rightarrow (c \bullet a \sqsubset c \bullet b) \wedge (a \bullet c \sqsubset b \bullet c)$.*

On the set $\mathbb{M} = N \times H \cup \{(\varepsilon, \infty), (\varepsilon, 0)\}$ we define addition and multiplication as above.

(a) The structure $S =_{df} (\mathbb{M}, +, (\varepsilon, \infty), \cdot, (\varepsilon, 0))$ forms an idempotent semiring, if $x \sqsubset x \bullet y$ for all $x, y \in H$; its natural order coincides on $N \times H$ with the lexicographical order $(n, y) \leq (m, x) \Leftrightarrow_{df} (x \sqsubset y) \vee (x = y \wedge m \preceq n)$; (ε, ∞) is the least and $(\varepsilon, 0)$ the greatest element.

(b) Under the conditions of Part (a) *and setting $r^* =_{df} (\varepsilon, 0)$ for all $r \in \mathbb{M}$ turns S into a Kleene algebra.*

Proof (sketch). The proof is by straightforward calculations. Parts of Part (a) were already stated and proved in [23].

(a) First, the lexicographical order of two total orders also forms a total order. Define (ε, ∞) as least and $(\varepsilon, 0)$ as greatest element, gives yet another total order, where addition calculates the maximum. Hence it is obvious that $(\mathbb{M}, +, (\varepsilon, \infty))$ forms a commutative monoid. Properties of $\cdot$ immediately follow by product construction (e.g. by Birkhoff's HSP theorem of universal algebra). Hence only distributivity has to checked by hand. Here the condition $x \sqsubset x \bullet y$ is needed.

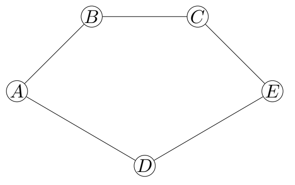
(b) Since in every Kleene algebra $1 \leq a^*$, defining Kleene star as $(\varepsilon, 0)$ is necessary. From this, the axioms can easily be checked. $\square$

Note that addition chooses the better route. Therefore the natural order reads as "worse than". In particular, $(n, y) \leq (m, x)$ means that the entry (n, y) is worse than (m, x). Most often this implies that (m, x) has a smaller hop count than (n, y).

For the concrete model used in the examples, the condition $x \sqsubset x + y$ obviously holds since $x, y \in \mathbb{N} - \{0\}$. Therefore the structure forms a Kleene algebra and axioms for finite iteration are available. The test algebra is discrete, i.e., it only consists of the two constants. The additional condition is not arbitrary, it even seems to be crucial for routing protocols since in [22] a similar property is used.

So far we only had a look at routing table entries. However, in general we are interested in sets of entries, where each route/entry also has a source and a destination. To model whole sets we use $n \times n$ matrices over entries. An entry at position i, j then corresponds to a route from i to j. To use matrices, we have to assume that the set of names is *finite*. In reality, this is not a restriction at all, since there can only be a finite number of nodes in a given network.[5] Each row of a routing table corresponds to a local routing table of a node. A routing table can therefore be seen as a "snapshot" of the whole network, where all node's routing tables are joined.

Example 5.4 Assume the connectivity graph depicted on the left hand side of Figure 4 and the arbitrary chosen routing table on the right.



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ A & (\varepsilon,0) & (B,1) & (B,2) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ B & (A,1) & (\varepsilon,0) & (C,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ C & (\varepsilon,\infty) & (B,1) & (\varepsilon,0) & (\varepsilon,\infty) & (E,1) \\ D & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (E,1) \\ E & (\varepsilon,\infty) & (\varepsilon,\infty) & (C,1) & (D,1) & (\varepsilon,0) \end{array}$$

Fig. 4. A simple network consisting of 5 nodes

Intuitively, the first entry $(\varepsilon, 0)$ states that if A wants to send something to A, there is nothing to do and the known path is "optimal", i.e., it has length 0; the entry $(B, 2)$ in the middle of the first row states that A knows a route to C with length 2. The next hop where A has to send the packet to is B. The entry (ε, ∞) indicates that no route is known. $\square$

In general a *routing table* is an element of the set $M(n, K)$ of all $n \times n$ matrices with elements in the Kleene algebra of routing tables presented before. Here, $n = |N|$ is given by the number of possible nodes.

Note that we restrict a routing table in no way. Therefore a node might not even know all its neighbours. Moreover, there is no need of symmetry, i.e., A knows a route to B does not imply necessarily that B knows a route to A. This reflects the reality. However, it seems reasonable to assume that a routing table has $(\varepsilon, 0)$-entries on its diagonal. This means that each node at the network knows at least itself. Algebraically this is easily expressed by $1 \leq a$.

As stated before tests are subidentities. In our matrix model a test is a matrix that has the entries (ε, ∞) or $(\varepsilon, 0)$ on its diagonal; all other entries of the matrix are (ε, ∞). Typically, a test is used to describe a set of nodes. Premultiplying an arbitrary matrix (routing table) by a test selects certain rows; hence the routing table information about particular nodes is selected. Postmultiplying selects columns, i.e., the snapshopt is "restricted" to information of a particular node or a set of nodes .

Looking at AODV, messages are sent throughout the network. Hence we have to define neighbours in a network, where messages can be sent to. Obviously, this is again done by a routing table.

[5] For AODV for example this number is bounded by 2^{32}, the number of possible unique IP addresses (`http://en.wikipedia.org/wiki/IP_address`).

A *topology* b is a special routing table which contains the identity as submatrix ($1 \leq b$) *and* also all available links between two nodes.

In our examples a topology contains all available one-hop connections.

Since we have shown that routing tables or, to be more precise sets of routing tables form well-known algebraic structures we are now ready to abstract to algebra and to model parts of AODV.

6 AODV Control Messages Algebraically

AODV is based on control messages sent through the network. As we will see it is less important whether messages are broadcast or unicast. At the algebraic level used, message sending can be expressed by

$$a + p \cdot b \cdot q \cdot (1 + c) \ , \tag{4}$$

where a, b, c are elements of an i-semiring S and $p, q \in \mathtt{test}(S)$. By distributivity, Expr. (4) consists of three parts: a, $p \cdot b \cdot q$ and $p \cdot b \cdot q \cdot c$. Informally, a describes the system before the message is sent (hence it is not part of the message itself); it can be seen as a snapshot which then, after the message has been delivered, is (if possible) updated by the two other summands. Remember that addition chooses the better route, hence it can be indeed interpreted as update. When a node p receives an AODV control packet from a neighbour q it creates or updates its routing table. If b represents the current topology, the equation $p \cdot b \cdot q$ establishes a 1-hop connection from p to q (if p and q are single nodes), i.e., p knows a path to q. If p, q are sets of nodes, only those paths between p and q are established that really exist in the topology. The third term ($p \cdot b \cdot q \cdot c$) transmits knowledge (encoded in an element or a routing table c) from q via the topology to p. In the meaning of routing table updates this again reads the other way round. Since p receives a message from q it can update its routing table with information of c.

Before modelling control messages we prove some useful properties about messages in general. For that we define a function

$$\mathtt{msg}(a, b, c) \ =_{df} \ a + b \cdot (1 + c)$$

for sending message c via b updating a; here b is either a given topology or it can be restricted by senders and receivers as before. That means that b might have the form $p \cdot b' \cdot q$.

Proposition 6.1.

(a) If the knowledge c and c' is fixed (does not change when sending a message), the order of sending does not matter, i.e.,

$$\mathtt{msg}(\mathtt{msg}(a, b, c), b', c') = \mathtt{msg}(\mathtt{msg}(a, b', c'), b, c) \ .$$

(b) If different messages are sent via a shared topology b, the messages can be sent in parallel, i.e.,

$$\mathtt{msg}(\mathtt{msg}(a, b, c), b, c') = \mathtt{msg}(a, b, c + c') \ .$$

(c) If the same message is sent via different connections, connections can be joined, i.e.,

$$\mathtt{msg}(\mathtt{msg}(a,b,c),b',c) = \mathtt{msg}(a,b+b',c)\ .$$

(d) Sending c via b using different sets of senders p and p' to receivers q and q' is not interchangeable. We only have

$$\mathtt{msg}(\mathtt{msg}(a,p\cdot b\cdot q,c),p'\cdot b\cdot q',c) \le \mathtt{msg}(a,(p+p')\cdot b\cdot(q+q'),c)\ .$$

(e) It is interchangeable if p is not connected to q' via b ($q' \le |b]\neg p$) and p' not to q, i.e.,

$$\begin{array}{c} q' \le |b]\neg p \wedge q \le |b]\neg p' \\ \Rightarrow \mathtt{msg}(\mathtt{msg}(a,p\cdot b\cdot q,c),p'\cdot b\cdot q',c) \ = \ \mathtt{msg}(\mathtt{msg}(a,p'\cdot b\cdot q',c),p\cdot b\cdot q,c)\ . \end{array}$$

Morover, if $q' \le |b]\neg p$ and $q \le |b]\neg p'$ then

$$\mathtt{msg}(\mathtt{msg}(a,p\cdot b\cdot q,c),p'\cdot b\cdot q',c) \ = \ \mathtt{msg}(\mathtt{msg}(a,p'\cdot b\cdot q',c),p\cdot b\cdot q,c)\ ,$$

meaning that sending can be done in parallel.

In general, Part (d) cannot be strengthened to an equation since p might be able to to send information to q' and p' to q. This behaviour is excluded when p is not connected to q' and p' not to q (cf. Part (e)).

The proofs are straightforward algebraic calculations. Moreover, they can be automated using first-order theorem provers—another advantage of an algebraic approach. We used Prover9 [16] to verify these properties. The input files can be found at [12].

A node following the rules of the AODV protocol often forwards messages. When a content c is forwarded, it is also changed. For example, if p receives knowledge c via b, it will not forward c but $b\cdot c$. Sending a message c via the topology b and forwarding it via another topology b' can be encoded by $\mathtt{msg}(\mathtt{msg}(a,b,c),b',b\cdot c)$. Assuming $1 \le b'$, we get

$$\begin{array}{cl} & \mathtt{msg}(\mathtt{msg}(a,b,c),b',b\cdot c) \\ = & a+b+b\cdot c+b'+b'\cdot b\cdot c \\ \le & a+b'+b'\cdot b+b'\cdot b\cdot c \\ = & a+b'(1+b+b\cdot c) \\ = & \mathtt{msg}(a,b',b+b\cdot c)\ . \end{array}$$

Intuitively, this means that the knowledge after forwarding a message once can be approximated by sending a single message via b' with knowledge of the first topology b and the learnt component $b\cdot c$. Forwarding and broadcasting a message through an entire network, is now forwarding the message again and again. In algebra this yields long, but simple expressions (cf. the calculation above). In the situation where the network topology does not change things become much easier. Forwarding once yields now the equation

$$\mathtt{msg}(\mathtt{msg}(a,b,c),b,b\cdot c) = \mathtt{msg}(a,b,b\cdot c)\ .$$

Hence, by simple fixpoints arguments, broadcasting a message can be modelled by (a single message)

$$\mathtt{msg}(a,b,b^*\cdot c) \;=\; a+b\cdot(1+b^*c) \;=\; a+b+b\cdot c+b\cdot b\cdot c+b\cdot b\cdot b\cdot c+\ldots\;.$$

The snapshot a is first updated with information from the topology b, then by information c sent via the topology by a 1-hop connection, then by the information c sent via a 2-hop connection and so on.

Before turning to the core messages of AODV, we look at a special case of forwarding messages. We assume that there is a sender p which broadcasts the empty message ($c = 1$) and that only those nodes that have actually received information forward the messages . The first message is given by $\mathtt{msg}(a, b\cdot p, 1) = a+b\cdot p$. The receivers of this message are characterised by $|b\rangle p$. After forwarding the received message once, the system looks like

$$\begin{array}{ll} & \mathtt{msg}(\mathtt{msg}(a,b\cdot p,1),b\cdot|b\rangle p,b\cdot p)\\ = & a+b\cdot p+b\cdot p+b\cdot|b\rangle p+b\cdot|b\rangle p\cdot b\cdot p\\ = & a+b\cdot(|b^0\rangle p+|b^1\rangle p)+b\cdot p+b\cdot b\cdot p\\ = & a+b\cdot(|b^0+b^1\rangle p)+b\cdot(p+b\cdot p)\;. \end{array}$$

The first step is by definition and distributivity. The second step holds since $p = |1\rangle p = |b^0\rangle p$, by commutativity, distributivity, and Equation (3). The third is again by distributivity and additivity of $|\_\rangle$. Propagating the message through the whole network can therefore again be expressed by using Kleene star.

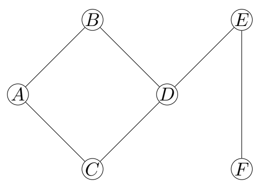
$$a+b\cdot|b^*\rangle p+b^*\cdot p\;. \tag{5}$$

Using the assumption $1 \le a$, this is equivalent to $\mathtt{msg}(a, b\cdot|b^*\rangle p, b^*\cdot p)$.

Let us now formalise the core messages of AODV.

Hello Messages do not send any information ($c = 1$) except the one of the connection. Moreover, if p sends such a message, every neighbour will receive it. Hence it has the form $a + b\cdot p$. In the matrix model terms like $b\cdot p$ restrict matrices column-wise. In particular this means that only information about routes towards p can be learned.

Example 6.1 Assume that the topology T is given by the graph of Figure 5; we further assume that D is the test element representing the single node D and that no node has any information about routes (except the trivial one to themselves), i.e., we start with the identity matrix as snapshot. Then $I + T\cdot D$ models that D sends a hello message. The result is given on the right hand side of Figure 5.
The resulting matrix shows that now the nodes B, C and E have established routes to D. $\square$



$$\begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ B & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\mathbf{D},\mathbf{1}) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ C & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\mathbf{D},\mathbf{1}) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ D & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ E & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\mathbf{D},\mathbf{1}) & (\varepsilon,0) & (\varepsilon,\infty) \\ F & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) \end{array}$$

Fig. 5. Sending Hello Messages

Route Requests are a bit more complicated. On the one hand a node that receives a message establishes a 1-hop connection as in the case of saying "Hello". This is the first iteration of the broadcast. However, if the neighbours do not have information about the destination, they forward the message by broadcasting it to their neighbours.

Example 6.2 Assume the topology T and the snapshot of Figure 5—denoted by X. Now we want to assume that A sends a RREQ looking for a path to D. As first step the snapshot is updated by $X + T \cdot A$, where A is the test representing the single node A; the result is

$$\begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ B & (\mathbf{A},\mathbf{1}) & (\varepsilon,0) & (\varepsilon,\infty) & (D,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ C & (\mathbf{A},\mathbf{1}) & (\varepsilon,\infty) & (\varepsilon,0) & (D,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ D & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ E & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (D,1) & (\varepsilon,0) & (\varepsilon,\infty) \\ F & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) \end{array}$$

We see that the nodes B and C have learned about the connection to A. Now these nodes should forward the request if they do not know D. However, both already know a connection to D, hence nothing happens.

Using I as snapshot instead of the one of Fiure. 5 yields another outcome.

$$\begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & (\varepsilon,0) & (B,1) & (C,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ B & (A,1) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ C & (A,1) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ D & (\mathbf{B},\mathbf{2}) & (B,1) & (C,1) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ E & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) \\ F & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) \end{array}$$

The original message is forwarded and broadcasted until it reaches D. □

Following the specification of AODV, the request is only forwarded if no route to the destination is known. In other words only those nodes have to forward the request who (at the beginning) do not have information about the destination q. To achieve this, we modify the topology b and use $b \cdot |a]\neg q$ instead.

Due to this forwarding and broadcasting a whole request can be modelled by

$$a + b' \cdot |b'^*\rangle p + b'^* \cdot p \ ,$$

where $b' = b \cdot |a]\neg q$.

Of course this compact algebraic expression is again only possible if the topology does not change. However, from a practical perspective it is reasonable to assume that the topology ceases to change after some time.

Route Reply For unicasting a route reply back, we restrict the topology or more precisely the snapshot in such a way that it contains only the single path back to the originator. At the moment we think that we need domain specific knowledge for this task.

Example 6.3 For the previously presented matrix, the path from D back to A can easily be constructed from the matrix: the information of D (the 4-th row) shows that the next hop towards A is B. A similar argument shows that the next hop from B towards A is A. Hence we would use a new topology that only contains these single 1-hop connections:

$$\begin{array}{c c}
 & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} &
\begin{pmatrix}
(\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\
(\mathbf{A},\mathbf{1}) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\
(\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\
(\varepsilon,\infty) & (\mathbf{B},\mathbf{1}) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\
(\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) \\
(\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0)
\end{pmatrix}
\end{array}$$

□

From an algebraic perspective, a route reply then becomes the same as a route request; just using the modified topology. In other words we restrict the topology to links that actually forward the request. In AODV each node that receives a route request from p and has information about a route to the destination q sends a reply. At least this set can be characterised purely algebraically by $|a\rangle q; |(b \cdot |a]\neg q)^*\rangle p$. The first part selects all nodes that have information about the destination q; the second one selects all nodes receiving information from p. Nodes only receive a route request if there is a path from p where no intermediate node has information about the destination.

Route Error messages are also spread throughout the network. Hence we can use the same mechanism of message sending discussed before. However, as a reaction of an incoming error message the routing table of a node has to be modified. More precisely, an entry has to be invalidated or removed. On the level of entries this can easily achieved by annihilation ($(m, x) \cdot (\varepsilon, \infty) = (\varepsilon, \infty)$). However, matrices select the best known route and any existing route would be preferred over (ε, ∞). There are two possible solutions, which we can only sketch (due to lack of space). One is to define a meet operation $\sqcap$ which selects the worst route. This would require a Boolean algebra as underlying structure. Another solution is to use some indication of freshness. In AODV this is achieved by sequence numbers (see Section 2). As soon as sequence numbers are embedded, invalidating route entries will come for free without changing the underlying algebraic structure.

To conclude this section, we show the feasibility of the algebraic approach and show an important property of the AODV protocol.

Theorem 6.4 *If a message (e.g. a route request) is broadcasted via a topology b with $1 \leq b$ and is not stopped (it is forwarded by all intermediate nodes), every node (connected to the originator) knows all its neighbours* and *each node knows a route to the originator of the message (if there is one). Mathematically this means for a topology b,*

$$q \leq |b^*\rangle p \ \Rightarrow\ b \cdot q + q \leq a + b \cdot |b^*\rangle p \cdot (1 + b^* p)\ .$$

The proof can again be automated; it takes less than a second.

In particular this means that the backward routes are loop free. The same argument then holds for the route request; hence whenever messages are forwarded through a topology (which might be changed for sending RREPs) the outcome is loop free.

However, as discussed in Section 2, loop freeness in general does not hold in our simplified setting. It holds in Theorem 6.4 since we assume that the message is broadcasted and forwarded throughout the entire network. To overcome this deficit, the protocol AODV provides sequence numbers to indicate the freshness of a route and information about validity of routes. Therefore a next step towards a real characterisation of AODV is to integrate sequence numbers in our algebra; however one crucial point is how to compose different routes (cf. Figure 3) with different sequence numbers.

7 Conclusion and Outlook

The aim of the present paper is to present first steps towards an algebraic characterisation for AODV. At the moment we are able to model the core parts of AODV like sending route requests. However the model excludes substantial details like sequence numbers.

The presented approach has several advantages: First, in its purely algebraic form simple algebraic calculations and reasoning with off-the-shelf theorem provers is feasible; secondly, on the model (matrix) level, standard and well established algorithms (like algorithms for matrix multiplication) can be used for model checking or for determining case studies[6]; thirdly, since in the model elements are routing tables, the connection between abstraction, “reality” and implementation (e.g., [1]) can be seen quite easily.

[6] In fact, we used a simple Haskell implementation to produce the example of the present paper.

The ultimate aim for future work is of course to completely specify AODV, to verify properties like loop freeness and maybe even improve the protocol. To achieve this goal, one has to add sequence numbers and route validity. However, when extending the underlying algebra the axioms of Kleene algebra should still be satisfied. In particular, the condition of Theorem 5.3 should hold. For reasoning about concurrency in AODV (e.g., when sending a message) one might extend the given algebra to a concurrent Kleene algebra [11].

So far we have modelled changing topology by separate matrices. Another approach would equip connections by probability. This would not only allow modelling changing topology, but also message losses during transmission. A first thought is to use probabilistic Kleene algebra [17, 24] instead of standard Kleene algebra. However as pointed out by Takai and Furusawa in the erratum to [24] probabilistic Kleene algebra is not closed under forming matrices.

A different approach to AODV might even be to look at the protocol from another point of view. In the present paper we used a classical approach; however if one interprets nodes as individuals or agents, their routing tables as the local knowledge, reactive protocols become multiagent systems where knowledge about routes is distributed. Hence one should be able to adapt yet another well-known theory to protocols.

Acknowledgement: We thank Peter Jipsen, Bernhard Möller and the anonymous referees for useful comments and suggesting improvements to the presentation.

References

1. AODV-UU: An implementation of the AODV routing protocol (IETF RFC 3561). `http://sourceforge.net/projects/aodvuu/` (accessed September 13, 2012)
2. Backhouse, R.: Closure Algorithms and the Star-Height Problem of Regular Languages. Ph.D. thesis, Imperial College, London (1975)
3. Backhouse, R., Carré, B.: Regular algebra applied to path-finding problems. Journal of the Institute of Mathematics and Applications (1975)
4. Carré, B.: Graphs and Networks. Oxford Applied Mathematics & Computing Science Series, Oxford University Press (1980)
5. Conway, J.H.: Regular Algebra and Finite Machines. Chapman & Hall (1971)
6. Desharnais, J., Möller, B., Struth, G.: Modal Kleene algebra and applications — A survey. Journal of Relational Methods in Computer Science 1, 93–131 (2004)
7. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
8. Fernandes, T., Desharnais, J.: Describing data flow analysis techniques with Kleene algebra. SCP 65, 173–194 (2007)
9. Griffin, T., Gurney, A.: Increasing bisemigroups and algebraic routing. In: Berghammer, R., Möller, B., Struth, G. (eds.) Relations and Kleene Algebra in Computer Science. LNCS, vol. 4988, pp. 123–137. Springer (2008)
10. Griffin, T., Sobrinho, J.: Metarouting. SIGCOMM Comp. Com.. Rev. 35, 1–12 (2005)

11. Hoare, C.A.R., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 09 — Concurrency Theory. LNCS, vol. 5710, pp. 399–414. Springer (2009)
12. Höfner, P.: Database for automated proofs of Kleene algebra, `http://www.kleenealgebra.de` (accessed September 13, 2012)
13. Höfner, P., Struth, G.: Automated reasoning in Kleene algebra. In: Pfennig, F. (ed.) Automated Deduction. LNAI, vol. 4603, pp. 279–294. Springer (2007)
14. Kozen, D.: The Design and Analysis of Algorithms. Springer (1991)
15. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Information and Computation 110(2), 366–390 (1994)
16. McCune, W.W.: Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9` (accessed September 13, 2012)
17. McIver, A.K., Gonzalia, C., Cohen, E., Morgan, C.C.: Using probabilistic Kleene algebra pKA for protocol verification. J. Logic and Algebraic Programming 76(1), 90–111 (2008)
18. Möller, B.: Dijkstra, Kleene, Knuth. Talk at WG2.1 Meeting (2006), slides available online at `http://web.comlab.ox.ac.uk/jeremy.gibbons/wg21/meeting61/MoellerDijkstra.pdf` (accessed September 13, 2012)
19. Möller, B., Struth, G.: WP is WLP. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) Relational Methods in Computer Science. LNCS, vol. 3929, pp. 200–211. Springer (2006)
20. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing. RFC 3561 (Experimental) (July 2003), `http://www.ietf.org/rfc/rfc3561.txt`
21. Singh, A., Ramakrishnan, C., Smolka, S.: A process calculus for mobile ad hoc networks. SCP 75, 440–469 (2010)
22. Sobrinho, J.: Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet. IEEE/ACM Trans. Networking 10(4), 541–550 (2002)
23. Sobrinho, J.: Network routing with path vector protocols: Theory and applications. In: Applications, technologies, architectures, and protocols for computer communications. pp. 49–60. SIGCOMM '03, ACM Press (2003)
24. Takai, T., Furusawa, H.: Monodic tree Kleene algebra. In: Schmidt, R.A. (ed.) Relations and Kleene Algebra in Computer Science. LNCS, vol. 4136, pp. 402–416. Springer (2006), Errata available at `http://www.sci.kagoshima-u.ac.jp/~furusawa/person/Papers/correct_monodic_kleene_algebra.pdf` (accessed September 13, 2012)