

# Beyond the Tree Structure: a new way to configure nodes in SONs

Julien Ridoux  
LIP6 - UPMC  
8, rue du C. Scott  
Paris, FRANCE  
julien.ridoux@lip6.fr

Anne Fladenmuller  
LIP6 - UPMC  
8, rue du C. Scott  
Paris, FRANCE  
anne.fladenmuller@lip6.fr

Yannis Viniotis  
ECE Department - NCSU  
Box 7911  
Raleigh, NC 27695, USA  
candice@eos.ncsu.edu \*

October 2004

## Abstract

Self-Organized Networks are multi-hop networks that do not rely on any infrastructure. Moreover, their topology is supposed to be flat since all nodes are equally in charge of address allocation and routing functions. Their time-changing topology breaks the association between Identification and Location that is present in the IP address. Moreover, mobility can not be handled easily when the addressing space is based on a tree description. In this paper we propose the use of Regular Virtual Structures to provide desired properties for Self-Organized Networks. Our approach opens new ways to build addressing spaces and enables an implementation based on trellis graphs. We show that the construction of the optimal trellises is an NP-Complete problem. We propose a heuristic and evaluate its performance via simulations. We also analyze the robustness of the proposed approach with regards to mobility.

## 1 Introduction

Wireless network technologies make the way users connect to any network seamless, and so allow them connectivity in almost any kind of environment. Disaster situations, battlefields, mass sport event meetings are examples where such connectivity is needed. Self-Organized Networks (SON) represent a network model proposed to support spontaneous and multi-hop networking without relying on any precise and *a priori* infrastructure. SONs are a general description unifying well-known research fields such as Ad-Hoc or Hybrid<sup>1</sup> networks, Peer-To-Peer systems and Sensor networks. A SON is then a spontaneous multi-hop network whose time changing topology depends on node arrivals in an environment where no infrastructure is present. By allowing a pervasive connectivity, SONs are a promising next step of the Internet evolution that has always been driven by the users' needs.

The spontaneous nature of SONs raises many new technical and research challenges. Routing, addressing, location management, name resolution, all are non-trivial issues because of the node mobility. We argue in this paper that the choice of the addressing scheme has a serious impact on all other problems. An address can have different purposes, depending on its level of abstraction. It can uniquely identify a node, inform on its localization, or it can also represent a combination of both. When it only identifies a host, the system has to ensure the uniqueness of each address in the network. In such a case, mobility has no influence on the value of the address. On the other hand, if position information is included in the address, the address has to be modified each time a node moves or appears in the network. This increases the cost of mechanisms used to disseminate or retrieve the association between nodes identifier and location.

---

\*The authors thank J-Y. Moulin, A. Soule, M. Latapy, D.N. Serpanos and the whole LIP6 team for their help and valuable comments on this work.

<sup>1</sup>We define hybrid networks as Ad-Hoc networks that can obtain an Internet connectivity with the aid of some of their nodes.

The complexity of routing issues is dependent on the localization information contained in the address. For geographical routing, the physical coordinates can easily help to define the path to reach a destination. Route determination is quite trivial in this case, but the complexity is transferred to the name resolution or location management process. For Peer-to-Peer solutions the localization is logical and another address space has to be used to route to the destination. IP addressing is certainly the most used and gives topological information on the node position in the Internet, information that can appear pointless if the node is not directly connected to the Internet. In the case of mobile networks such as Ad-Hoc networks, this topological information loses its meaning. This leads to a situation where the IP address only has an identification purpose and where routing does not scale easily.

The technical problem we address in this paper concerns both address allocation and routing in a SON. Address allocation has to be distributed because of the lack of infrastructure, which requires complex mechanisms with a substantial overhead on the network. The address allocation and routing problems in a SON can then be seen as a generic minimum cost query problem to execute on an infrastructure-less environment. With this point of view, nodes request available addresses, routes to destinations and resources from the other participants. The envisioned solutions to this “query problem” have been handled differently depending on which context they have been proposed, as we describe next.

A large suite of Ad-Hoc networks protocols based on IP addressing have been designed to allow a dynamic auto-configuration of nodes and an implied routing protocol definition. In a general manner, they address the query problem with a flooding mechanism. The address allocation is usually based on a Duplicate Address Detection (DAD) mechanism, while route discovery relies on improved flooding requests.

Geographical routing [1] answers the route query problem thanks to the geographic coordinates of the nodes. Their identification is decorrelated from their location that implies need of a location mechanism to realize the address query such as GLS (Grid Location Service) [2] and to retrieve the position of the destination.

Peer-To-Peer systems [3], [4] define a virtual addressing space based on Distributed Hash Tables. This virtual space realizes the correspondence between the node identifier and their location in the virtual space. The routing query problem is solved by the underlying layer usually based on an infrastructure. To push the Peer-To-Peer concepts in to the routing layer, approaches such as [5], [6] propose the use of a virtual structure to ensure routing by themselves. These addressing spaces are based on a tree structure, that copes poorly with the dynamic nature of SON.

IP-based solutions for Ad-Hoc networks require flooding to solve the addressing and routing query problems. GLS and the Peer-To-Peer approaches solve the addressing query problem by introducing a virtual structure. These examples show that the cost of solutions to the query problem decreases when a structure is present. The study done by Castro *and al.* in [7] is a formal description showing that structured networks are helpful to introduce a coherent addressing scheme and possess good characteristics in terms of routing and maintenance performances. In this paper we claim that the use of a virtual structure will give topological properties to SONs that they do not naturally have.

An example of the use of Virtual Regular Structures (VRS) has been studied in the context of geographical routing, for location purposes with the GLS [2]. In order to allow the nodes’ location information to be distributed among the nodes, GLS imposes a VRS known a priori by all nodes and dividing the geographic space into a hierarchy of grids. With this predefined structure, GLS provides a deterministic method for the nodes to store and retrieve location information but its performance varies with the geographical distribution of nodes.

Our proposal can be summarized as follows. In order not to be constrained by the geographical density of nodes, we provide a topological VRS to map the addressing space. This topological description will give a direct application for routing that needs a topological knowledge of the network. The basic element we use to propose an implementation of the VRS is a trellis graph. Trellises can be used for both creating the address space and finding routing paths. They provide an implicit multi-path construction and robustness toward frequent node arrival and departures, because of their redundant structure.

The key points of our proposal are defined in terms of addressing and routing in SONs. We design an addressing space that is set up in a “constructive” and local manner. This local construction avoids the need of a global view

of the network that is difficult to obtain in a SON. Such a construction allows the addressing space to adapt to the number of nodes present in the SON. The more nodes that arrive in the network, the bigger the addressing space is. This mechanism is only limited by the capacity of nodes to manage large addresses. The dynamic address allocation mechanism we provide generates addresses that are relative to the set of existing ones. This avoids address allocation conflicts and is realized in a local manner. This mechanism avoids widespread flooding on the network while ensuring the uniqueness of the addresses.

In terms of routing, our virtual structure includes redundancy that provides an implicit multi-path definition that is an essential property for SONs. Moreover, the type of graphs we use can be described in an algebraic way. This description is a powerful tool to compute routes locally and so makes our routing scheme scalable, by avoiding large flooding for route discovery.

In the rest of the paper, section 2 details requirements to design addressing and routing for SONs. Section 3 describes in a general fashion the use of regular structures and the trellis structure we chose to implement them and to answer the generic query problem. Sections 4 and 5 detail the addressing space and the corresponding routing mechanisms with regards to our implementation. We prove that the trellis construction problem is NP-Complete and provide a heuristic solution. Section 6 contains analysis in terms of robustness of our approach in the presence of mobility. Section 7 presents simulations showing the feasibility of constructing a regular VRS based on topological information.

## 2 Design Requirements for Addressing and Routing in SON

We propose eight requirements in order to compare addressing and routing solutions for SON. This list is not meant to be exhaustive; it simply provides a common basis for comparison.

- The dynamic address allocation must be realized without the aid of any existing infrastructure.
- The complexity of the addressing space management must be low when nodes join, leave or move.
- Both routing and addressing allocation must be realized in a distributed and fair manner (helping network mergers and splits).
- Both addressing and routing must scale.
- Addressing and routing must be robust toward mobility and errors in location estimation.
- Addressing and routing must limit control message overhead.
- Routes must be discovered or computed easily.
- Routing process has to produce a “fair” allocation of the routing overhead to nodes.

In the following we use these requirements to compare existing approaches.

The well-known IP addressing should ideally be described hierarchically. In its “classical” use, an IP address combines both identification and location information of a node. This allows route aggregations, and good scalability of the routing protocol in use. To exploit this property, the physical infrastructure of wired IP networks has been built mainly in a hierarchical manner. In the case of a SON, the reasoning to set up the network is the exact opposite. Nodes form an unknown and non-hierarchical topology because of their arrival process. On such a topology, only the identification information contained in the IP address makes sense. Decorrelating the node identifier and location into two different addresses allows SONs to deal with mobility and helps avoid flooding.

While they provide a straightforward compatibility with the existing IP network, IP-based solutions do not decorrelate identification and location of nodes in the address and then can not provide robustness toward mobility. The IP address is only used as an identifier. References [8] and [9] describe approaches that rely on an infrastructure to realize the dynamic address allocation and can only be applied to hybrid networks. References [10], [11], [12],

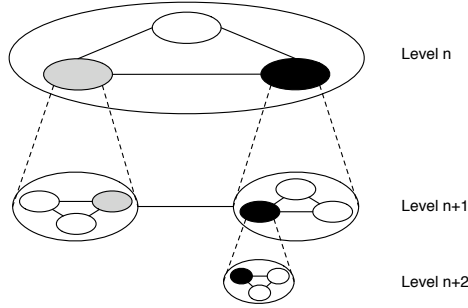


Figure 1: A recursive regular structure.

[13], [14] provide a distributed address allocation but use a DAD and the implicit flooding to ensure the uniqueness of each IP address. The DAD mechanism is not totally reliable and leaves possible address conflicts. References [15], [16] provide distributed routing algorithms but generate a high overhead of control messages, especially when nodes are highly mobile. Under these conditions more frequent flooding is required for DAD and route discovery that may prevent them from scaling.

By assuming nodes to have a Universal Identifier (UI), geographical approaches get rid of possible address conflicts and propose a distributed and easily manageable geographic addressing space. Addressing and routing possess excellent scalability properties, are robust toward mobility and compute routes easily thanks to the geographic position associated with nodes. Their main drawbacks are the need of a positioning system and their location mechanism to retrieve the position of the destination. Since the density of nodes in the network can be sparse, some of the nodes running GLS may have to maintain a high volume of node location information.

Peer-To-Peer systems propose a distributed addressing space decorrelated from node identifiers. Their addressing structure allows scalability but is mainly static. They handle mobility and frequent join/leave procedures with difficulty. An efficient management of the addressing space becomes then difficult to realize efficiently. Alternative approaches such as [3] and [4] rely on an underlying infrastructure and have addressing spaces difficult to manage node mobility. Proposals such as [5], [6] free themselves from the need of an underlying infrastructure. But the tree structure they propose and the addressing space used to route make it difficult to manage the addressing space in the presence of mobility. The following section describes the addressing structure we propose and the resulting routing operations.

### 3 Generic Virtual Structure

#### 3.1 Generic Description

In order to provide a virtual structure and decrease the cost of the generic query problem in a SON, we propose the use of a regular structure based on a topological knowledge of the network. This regular structure has to possess properties that facilitate addressing and routing. This regular structure is repeated to integrate all the nodes of a SON. In order to allow a communication between these regular structures, they have to be connected to each other. To keep the same properties induced by the regular structure, we propose to connect them in a recursive manner. Some of the nodes act at different levels of recursion and provide the connectivity between those levels. These nodes are called “Structure Heads”, represented in grey color in figure 1 (Figure 1 will be explained in more detail in section 4). The construction of the virtual structure is optimized by mapping a maximum number of nodes to it.

Routing in such an organization of regular structure implies some requirements:

- Each regular structure must possess an address that identifies it and gives the recursion level in which it is present.
- Each node of a regular structure must possess at least one address relative to the structure.

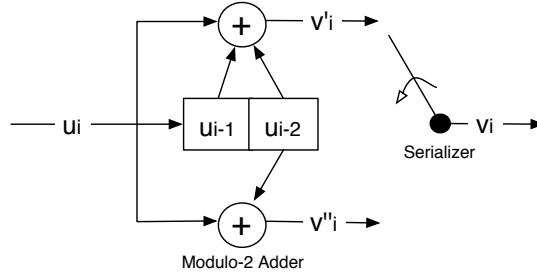


Figure 2: A (2,1,2) Convolutional Encoder.

- All nodes in a regular structure must be able to communicate to each other, through a path in the physical layout.

The address of a destination contains the path to reach it in the whole virtual structure. This path is composed of the list of Structure Heads of each regular structure a packet has to go through. Locally to a structure, the routing decision is taken compared to the address of the next Structure Head. The knowledge of which regular structure to forward packets to is given by the address identifying each regular structure. In order to realize this routing mechanism, each link present in the virtual structure must correspond to a physical one.

### 3.2 Realization based on Trellis Graphs

In order to implement the general concepts we just described, we chose to use trellis graphs because of their redundant and algebraic characteristics. To the best of our knowledge, trellis graphs have been used only in [17] at the networking level. Their application in providing a VRS is novel. A trellis graph can be generated by a Convolutional Encoder. Convolutional Encoders [18] are binary encoding mechanisms with memory that are usually defined as a sequential circuit. In communication systems, they introduce redundancy to the input information stream to allow error recovery after transmission on a communication channel. A convolutional code is defined by three parameters  $(n, k, L)$ .  $n$  corresponds to the number of outputs to the code for a given number of  $k$  entries. A convolutional code is composed of a shift-register of  $L$  stages, one or several modulo-2 adders and a multiplexer for serializing the encoder outputs. Since a Convolutional Encoder has  $L$  unit-delay cells, a given output  $v_i$  depends on the former inputs  $u_i, u_{i-1}, \dots, u_{i-L}$ . A (2,1,2) convolutional encoder is illustrated on figure 2. A Convolutional Encoder can be represented by a Finite State Machine (FSM) or a trellis graph. Figure 3 illustrates the representation of the (2,1,2) Convolutional Code we described.

The FSM states are labeled by binary values indicating the current bits stored in the encoder memory. The transitions are labeled by binary values (*e.g.*, 0/11) indicating the entry given to the convolutional encoder (0) and the corresponding output code (11). The same information is represented on the corresponding trellis. From the state 00, the FSM changes to the state 10 if the new input value to the encoder is 1 and stays in the state 00 if the input value is 0. Words are encoded by reading them in Least Significant Bits order.

A trellis is a connected graph, repeating a set of vertices, which all have the same degree. The vertices of the trellis represent the different FSM states and their edges the FSM transitions. The representation of the FSM state changes for a given input sequence corresponds to a path in the trellis. Choosing a particular Convolutional Encoder defines the size of the corresponding trellis and the degree of its vertices. In the following we use the FSM and the trellis illustrated by figure 3 to detail the mechanism used for addressing and routing.

## 4 Virtual Address Space based on Trellises

In order to precisely describe the way we organize the Virtual Regular Structure, we state some hypotheses first.

- We consider a Self-Organized Network built from scratch, where node arrivals are sequential. The physical topology created by this arrival process can not be predicted and we assume no particular property on it.

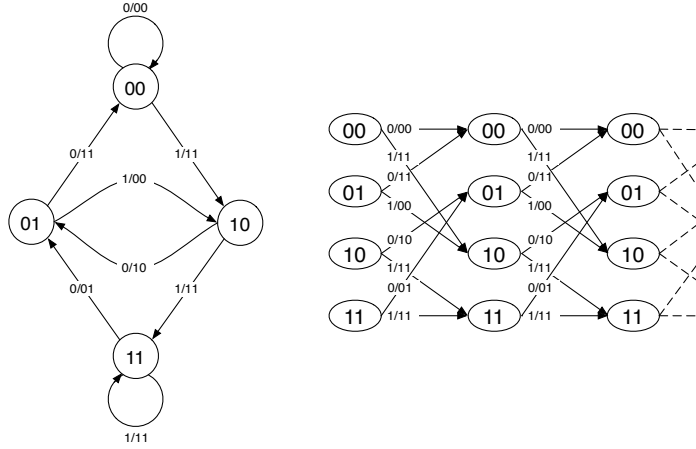


Figure 3: FSM and Trellis representations of a Convolutional Encoder.

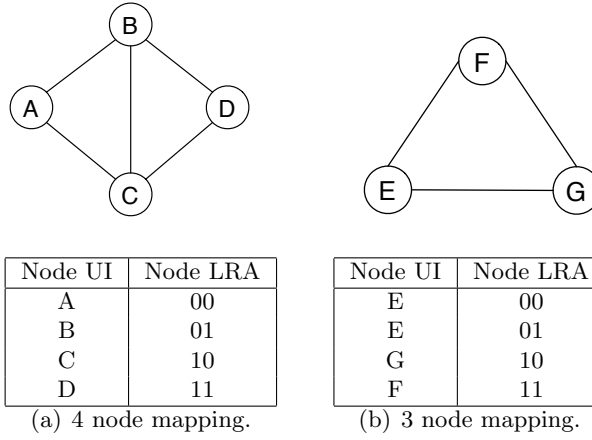


Figure 4: Basic Physical Topologies and Association Tables.

- We suppose that no infrastructure is present either, to help the addressing or routing mechanisms<sup>2</sup>.
- We make the hypothesis that each node joining the network possesses a unique Universal Identifier (UI). The definition of this identifier is outside the scope of this paper; the reader can refer to the large amount of literature in different fields such as Peer-To-Peer systems [3], [4]. To allow compatibility with the existing IP network, an IP address could be used as the node UI, as long it is guaranteed to be unique.

The key point of the trellis-based VRS construction is the mapping of the physical nodes Universal Identifier to the states of a FSM, *i.e.*, vertices of a trellis.

#### 4.1 Description of a Single Trellis

The vertices of a trellis graph represent the location of nodes within the virtual structure we want to set up. The FSM in use to generate the trellis, is known *a priori* by all the nodes composing the network. The state of the FSM associated to a node is called its Local Relative Address (LRA). Since the identifier and location of a node are decorrelated, we need to introduce an address Association Table to realize the mapping of the node UI to its Local Relative Address. Figure 4 gives two simple physical network topologies to illustrate this mapping where capital letters represent the nodes UI. The topology of the figure 4(a) is composed of four nodes, which are associated to the FSM states as shown in the corresponding table. In this case, each node of the physical topology is associated

<sup>2</sup>An existing infrastructure can be present and used by our VRS in a transparent way, and will ease the construction of the VRS. Here we only consider the general case where no infrastructure is present.

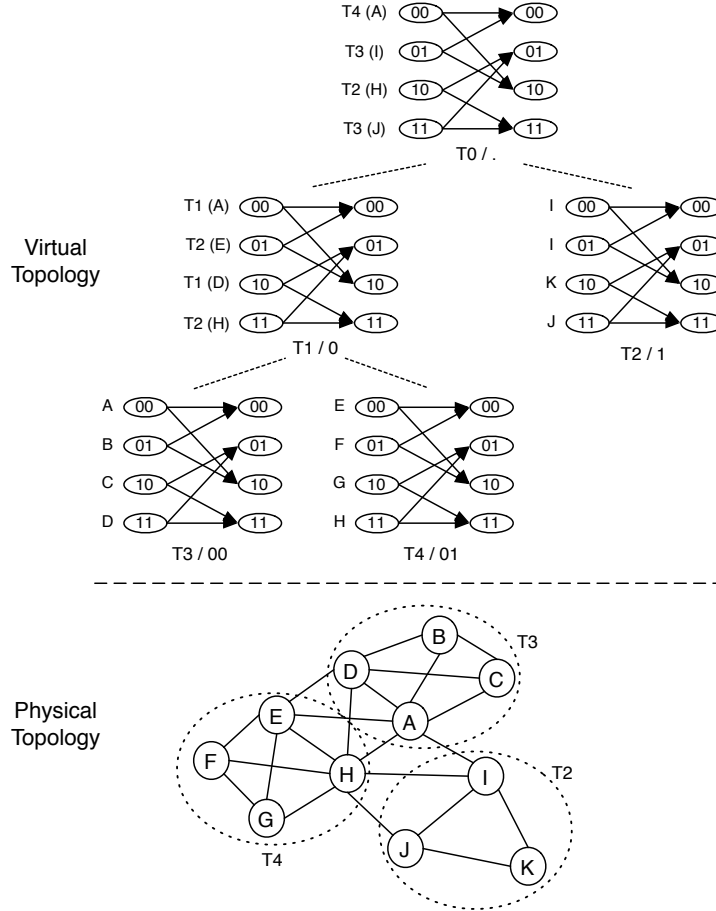


Figure 5: Spanning the Network.

to a unique state of the FSM. Figure 4(b) represents a three node topology. In this second case, since the number of physical nodes is lower than the number of states of the FSM, the node  $E$  is represented twice, *i.e.*,  $E$  is being given two addresses in the regular structure. As it is shown in this example, each trellis possesses a given number of addresses and one or several of these addresses are allocated to a physical node. The resulting Association Table representing neighbor nodes depends on their arrivals. Different node arrival sequences, resulting in the same physical topology could produce different node UI to FSM states mappings, and different numbers of Local Relative Addresses of a node. The Association Table creation is detailed in section 4.4 with the description of the construction heuristic. In the examples given in figure 4, the paths to nodes are their LRA. If node  $A$  needs to communicate with node  $D$ , it will use  $D$ 's LRA (11). By construction, each edge of the trellis corresponds to an existing physical link.  $A$  can then find a path in the trellis to reach  $D$ . The routing process will be described precisely in section 5.

## 4.2 Spanning the entire Network

Recursion is introduced by creating trellises of trellises. This recursive construction is our proposed solution to span the entire network by repeating the same regular structure. Figure 5 shows trellis graphs  $T_3$ ,  $T_4$  and  $T_2$  mapping respectively the nodes  $\{A, B, C, D\}$ ,  $\{E, F, G, H\}$  and  $\{I, J, K\}$ .

In order to allow all the nodes of the network to communicate, the trellis graphs  $T_2$ ,  $T_3$  and  $T_4$  have to be interconnected. This connection is realized in a recursive manner. Figure 5 gives an example of a VRS with several trellises and two levels of recursion. As an example,  $T_1$  associates  $T_3$  to its states 00 and 10, while it associates  $T_4$  to its states 01 and 11.  $T_0$  and  $T_1$  can then be described as “trellises of trellis graphs”.

The VRS structure we describe aims to provide both an addressing space and a routing protocol. But trellis graphs belong to the virtual space. To communicate from one trellis to another, the communication has to be ensured by the physical nodes interconnecting the two trellises. To realize all the operations our mechanism needs, trellis graphs are represented in higher levels of recursion by some of the physical nodes that permit this interconnection. The two nodes chosen to represent their trellis in higher recursive level are those associated to the lowest and to the highest value of the FSM states. We name these two nodes the Trellis Heads since they represent the trellis in higher levels of recursion. In the example of figure 5,  $T_3$  is represented in  $T_1$  by its Trellis Heads  $A$  and  $D$ . In the same manner,  $T_4$  is represented by nodes  $E$  and  $H$  in  $T_1$ .

As we mention in section 3, we need to define a Trellis Prefix (TP) to identify each regular structure. By definition each trellis of size  $2^L$  gathers  $2^{(L-1)}$  sub-trellis. Each trellis  $T$  attributes a prefix  $p$  written as a bit string of size  $2^{(L-1)}$  to the nodes representing a common sub-trellis.  $p$  is the recursive concatenation of  $T$ 's TP with the bit string attributed by  $T$  to a sub-trellis. The highest trellis in the recursive hierarchy possesses a prefix of size 0. An example of the resulting TP attribution is illustrated in figure 5, where  $T_4$ 's TP is 01, the concatenation of  $T_1$ 's TP and  $T_4$ 's prefix in  $T_1$ .

Once the trellis structure is established, all the states of each trellis composing the VRS are associated to a physical node. It is then possible for a node to retrieve the path to any destination in the VRS. This path, called the Relative Address (RA), is a bit string composed of the concatenation of Local Relative Addresses. For each trellis a packet goes through to reach the destination, one LRA is concatenated to the Relative Address. The LRA added in each trellis is the LRA of the last node the packet goes through before being forwarded in another trellis. The RA is the sequence of "exit vertices" of each trellis to go through to reach the destination. Since the RA corresponds to a path, a destination RA is different for two distinct source nodes.

As an example, consider  $K$ 's Relative Address in figure 5. Let suppose that  $F$  and  $J$  need to reach  $K$ . For  $J$ , the case is trivial.  $J$  and  $K$  belong to the same trellis, they share the same Association Table, and  $J$  knows  $K$ 's RA as 10. From  $F$ 's point of view,  $K$ 's RA will be 00.11.11.10. That means  $E$ 's LRA in  $T_4$  (00),  $H$ 's LRA in  $T_1$  (11),  $J$ 's LRA in  $T_0$  (11),  $K$ 's LRA in  $T_2$  (10). It is important to notice, that this is not the only possible RA for  $K$  from  $F$ 's point of view. Since we introduce redundancy for multi-path properties, and because the RA is bound to the path, 11.00.01.10 would have been, for example, another possible one.

### 4.3 Trellis Construction Complexity

Establishing an optimized trellis-based VRS as we just described is the biggest difficulty in our approach. In the following we show that for a given network topology, the set up of the optimal VRS is NP-Complete. In order to prove it, we formalize the problem in terms of graph theory and show the equivalence with the well-known H-Matching problem. The trellis graphs belong to the virtual space and their representation on the network corresponds to a connected subgraph of the physical network topology. We name this connected subgraph, representing the FSM and the Association Table representing physical nodes, the FSM Derived Graph.

Let  $\mathcal{M}$  be a FSM with a number  $s$  of states  $s_i$ .  $\mathcal{M}$  can be represented as an edge-labeled, directed graph  $M$ . We first make the adjacency matrix of  $M$  be symmetric and put the values on the diagonal to 0 (to remove the self loops). The resulting graph  $M'$  is then a connected undirected graph without loops.

Let  $W$  be a set of vertices such that  $|W| \in [2, s]$ .  $W$  represents the nodes of the physical topology belonging to a common Association Table. By construction of our scheme, the Association Table is meaningful only if it represents at least two physical nodes. Then,  $\forall i \in [1, |W|], \forall j \in [1, s]$ , there exists an application  $\mathcal{A}$  such that  $v_i \mathcal{A} s_j$ .  $\mathcal{A}$  corresponds to the association between physical nodes and FSM states recorded in the Association Table of the trellis and creates an instance of  $M'$ .

By removing possible duplicate entries in the  $M'$  adjacency matrix, we obtain one of the FSM-derived graphs  $D = (W, F)$ . (The set of edges  $F$  is defined by  $D$ 's adjacency matrix). By construction,  $D$  is a connected graph whose cardinality depends on the Association Table. Figure 6 gives the possible Derived Graphs of the FSM represented in figure 3, where  $|W| \in [1, s]$ .

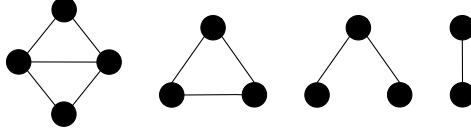


Figure 6: The four possible Derived Graphs of the figure 3 FSM.

Let  $G = (V, E)$  be the undirected, connected graph representing a given network topology. Let  $D = (W, F)$  be one of the possible connected, derived graphs of a given FSM  $\mathcal{M}$ . Since the trellis graphs belong to the virtual topology, establishing the trellis graphs that represent the physical nodes is equivalent to realizing a H-Matching of  $G$ , where  $H$  is the Derived Graph  $D$ . Then,

- (i) if  $|W| = 2$ , the H-Matching of  $G$  by  $D$  is realized in a polynomial time.
- (ii) if  $|W| = 3$ , the H-Matching of  $G$  by  $D$  is NP-Complete. The proof of the NP-Complete complexity of such an H-Matching can be found in [19].

The smallest FSM that can be generated by a Convolutional Encoder possesses two states and so produces a unique Derived Graph consisting of two connected vertices. The FSMs produced by more complex Convolutional Encoders, that are of more interest, induce a set of derived graphs. Obtaining an optimal H-Matching with the set of a FSM's Derived Graphs implies placing as many nodes as possible in a trellis, *i.e.*, using the Derived Graphs for which  $|W|$  is maximal. Establishing an optimal trellis set gathering the physical nodes is then NP-Complete.

As we said, setting up the whole VRS on the topology represented by graph  $G$  is a recursive mechanism. To set up the trellis of trellis, the construction process must respect a constraint: nodes representing trellis graphs in any recursive level must be able to communicate physically. That is why each trellis edge of the VRS must correspond to a physical link of the network topology. The choice of the Trellis Heads at each recursive level is then important. We have to select the two Trellis Heads  $u_{G_i}, v_{G_i}$  in each collection of disjoint graphs of  $G$ ,  $G_i \dots G_k$ , produced by the H-Matching. This selection procedure  $\mathcal{P}_S$  is done such that :

- (i)  $G' = (V', E')$  with  $V' = \bigcup_i \{u_{G_i}, v_{G_i}\}$  and  $E' \subset E$  is the set of edges connecting any two distinct nodes  $u, v \in V'$ .
- (ii) The graph  $G' = (V', E')$  is a connected graph.

The same H-Matching based on the Derived Graphs set has to be applied on  $G'$ , with the same optimal criterion as associating as many nodes of  $V'$  as possible in each Association Table. Even if  $\mathcal{P}_S$  can be realized in a polynomial time, it needs the result of an NP-Complete function realizing the H-Matching at the lower recursion level. This is why finding the optimal trellis-based VRS on a given topology is an NP-Complete problem.

#### 4.4 Heuristic for Trellis Construction

Since the construction of the VRS is an NP-Complete problem, we propose a heuristic to set it up, based on the fact that nodes join a SON in an incremental manner. In this way, our heuristic builds the top trellis of the VRS first and adds new, lower level trellises as nodes join the SON. We describe in the following algorithm 1 the key points of the heuristic only. When a node joins a network, it tries to join the existing VRS by being inserted in an existing trellis. If no possible position is available, because of physical connectivity constraint, the arriving node creates a sub-trellis with its neighbor. This process ensures that the heuristic is able to configure any new node joining the network.

Compared to the requirements we described in section 2, we showed in this section 4 that our scheme allows a dynamic address allocation without the help of any dedicated infrastructure. The recursive and constructive establishment of our VRS allows a possibly infinite quantity of available addresses to new nodes. Our addressing space is managed in a distributed (and local) manner among physical neighbor nodes without using flooding mechanisms, by the use of a simple Association Table. This solution allows us to solve the addressing query problem with a very low cost.

---

**Algorithm 1** Construction Heuristic

---

```
node  $N$  joins the network
if  $N$  detects some neighbors then
  each neighbor returns the list of trellises it belongs to
   $N$  sorts the trellises by decreasing number of nodes they contain
  for each trellis do
     $N$  lists the available positions in the trellis
    for each position do
       $N$  checks recursively the position in lower trellis
      if the position is one of the Trellis Head then
         $N$  checks recursively the position in upper trellis
      end if
      if position is valid then
        break
      end if
    end for
  end for
if No available position has been found then
  for each trellis do
     $N$  selects two neighbors that do not belong to a lower trellis
     $N$  creates a sub-trellis with these two neighbors
  end for
end if
else
   $N$  creates a new trellis
end if
```

---

SrcUI	DstUI	RAsize	DstRA	TPsize	DstTP
-------	-------	--------	-------	--------	-------

Figure 7: Necessary Packet Header Information.

## 5 Packet Forwarding using the Trellis Structure

In this section we describe the basic routing operations nodes have to implement. We describe a node position retrieval mechanism that is the most natural compared to our structure, even though it may not be the most efficient. Other approaches can be studied, such as the use of Distributed Hash Tables in order to optimize this process. We will not detail these different optimizations as it is not the core of our paper.

### 5.1 Node Location Registration

We have two extreme solutions to achieve RA retrieval, namely a proactive and a reactive one. Since we define a virtual structure, we are able to define a tradeoff between these two extreme possibilities. Our scheme is based on a half-proactive, half-reactive approach. The RA of a destination is built during each part of this process.

All node information in a regular structure is maintained by their Trellis Heads. This information is passed and maintained recursively by all Trellis Heads, until reaching the root of the recursive tree. This process is triggered proactively each time an Association Table is modified, as nodes join, leave or move. Before communicating their Association Table to Trellis Heads of  $T_{n-1}$ , Trellis heads of  $T_n$  concatenate their LRA in  $T_{n-1}$  to the information they maintain. This process ensures that each Trellis Head knows the routing information of all nodes of the trellis graphs below it. Moreover, this mechanism produces the bit string corresponding to the path from the Trellis Head to the destination node.

### 5.2 Operations at Source

For simplicity of presentation, we consider the scenario, illustrated in figure 5, where node  $F$  needs to send a packet to node  $K$ . The way  $F$  retrieves  $K$ 's Universal Identifier is out of the scope of this paper, we then make the assumption  $F$  has a way to know  $K$ 's UI. In order to initiate the communication,  $F$  has to be aware of  $K$ 's Relative Address and  $K$ 's TP. To retrieve the path to  $K$ ,  $F$  has to reach the first Trellis Head that knows  $K$ 's position.  $F$  sends a request to one of its Trellis Heads, for example  $E$ . If  $E$  knows  $K$ 's RA, it concatenates its own LRA to  $K$ 's RA and sends it back to  $F$  with  $K$ 's TP. If  $E$  does not know  $K$ 's RA, it sends a request for  $K$ 's routing information on its own, to one of its higher recursion level Trellis Heads. This process is repeated recursively until the request reaches a Trellis Head that knows  $K$ 's routing information. If  $K$  is present in the network, the process will obviously find an answer (in the worst case from a node present in the top-level trellis). When  $F$  retrieves the answer to its request, all Trellis Heads the data packet will go through would have added their LRA, forming the complete path to the destination.

Suppose next that  $F$  retrieved  $K$ 's Relative Address.  $F$  can now initiate a communication with  $K$ . For this purpose,  $F$  has to place information in the packet header. Figure 7 indicates the necessary fields of the packet header the source has to fill. In our example,  $F$  has to fill the first field with its own UI, and the second with  $K$ 's UI. Since the size of  $K$ 's RA is different for any source node, the third field is used to define  $K$ 's RA size. The fourth field indicates  $K$ 's RA. The fifth and sixth fields indicate the size and  $K$ 's TP. As soon as the source node  $F$  placed all the information in the packet, it can send it on the network in the same way as intermediate nodes. This process is described in the following section 5.3.

### 5.3 Operations at Intermediate Nodes

Routing is based on the Virtual Regular Structure. Since edges of trellis graphs correspond to physical links in the topology, finding a path in the trellis graphs determines a route in the physical topology. Moreover, finding a path in a trellis corresponds to encoding the label of the destination vertex one wants to reach. When a relay

node  $R$  receives a packet, it checks if it is the final destination the packet is intended to, by comparing the  $DstUI$  field of the packet header to its own UI. If  $R$  is not the destination of the packet, it forwards it.

If  $R$  has to forward a packet, it first has to know in which trellis it has to relay the packet in. To make this decision,  $R$  performs a comparison between its own Trellis Prefix and the  $DstTP$  field of the packet header. The decision process follows the algorithm 2. Since  $R$  can be present in different recursive trellises, algorithm 2 ensures the packet to be forwarded correctly along the trellis structure.

---

#### Algorithm 2 Trellis Selection

---

```

if  $size(DstTP) < size(R's\ smallest\ TP)$  then
   $R$  selects its trellis whose TP is the smallest
else
   $m = prfx\_match(DstTP, R's\ longest\ TP)$ 
  if  $size(m) == 0$  then
     $R$  selects its trellis whose TP is the smallest
  else
     $R$  selects the trellis  $T$  such that  $size(T's\ TP) == m$ 
  end if
end if

```

---

Once  $R$  knows in which trellis it has to relay the packet, it has to decide which node will be the next hop. If the prefix matching comparison did not give any common bit string, the packet is on the way up to the top of the VRS; otherwise the packet is being relayed on the way down to the destination. The following algorithm 3 describes the operations realized by  $R$  on the different bit strings.

---

#### Algorithm 3 Bit String Extraction

---

```

TopSize = RAsize - L x TPsize;
if  $size(prfx\_match(DstTP, currentTP)) \neq 0$  then
  start = TopSize - L x CurrentTP + 1;
else
  start = TopSize + L x CurrentTP + 1;
end if

```

---

$R$  extracts the  $b_i$  bits of DstRA such that  $b_i \in [start, start + L]$  (DstRA first bit is numbered as 1). The initial state of the FSM is set as the value of  $R$ 's LRA in the current trellis. This encoding process gives  $R$  the complete path along the current trellis the packet has to be relayed. Following strictly the path given by this sequence would make the routing be really far from the optimal physical shortest path. In order to optimize the routing process, the routing protocol takes into account the physical neighborhood of nodes. To realize this,  $R$  selects as the next hop, its last physical neighbor present in the computed path. This allows a local optimization of the routing process within the trellis and an implicit multi-path scheme as the encoding process gives a *list* of possible next hops, instead of only a single one.

## 5.4 Features of our approach

Our scheme proposes to retrieve the position of nodes, and thus the corresponding routes, thanks to a query mechanism that avoids flooding. But this process is unfair compared to the dissemination of nodes location information. Indeed, nodes present at higher levels of trellis graphs store more information than lower level nodes. This is a drawback. The worst case concerns the nodes being at the highest trellis of the VRS, since they have to maintain a state for each node participating in the network.

In terms of routing, the trellis structure induced by the FSM allows a simple computation of paths within a trellis by a bit string encoding that is realized in linear time, and make it scalable. In term of routing strategy, our VRS produces a built-in multi-path routing scheme. This allows the construction of alternative routes and

increases robustness of routing. As we mentioned earlier, our routing path is not optimal in terms of path length. Our next hop selection process ensures a local optimal shortest path in a trellis, but not an end-to-end shortest path. Nevertheless, the concatenation of local shortest path leads to nearly the end-to-end shortest path. In section 7 we evaluate via simulations the performance of our heuristic approach.

## 6 Robustness Analysis

Mobility is arguably the biggest challenge SONS have to face in regard to addressing and routing. Our approach decorrelates node identifier and position, which provides the basis to handle mobility. Note that mobility takes on several aspects that have to be managed in different ways. None of the solutions developed so far for SON are able to handle all aspects of mobility without severe drawbacks such as high control overhead, or addressing space reallocation.

In order to position our work toward the different mobility aspects, we define four mobility categories. The first one is the join and leave processes of nodes. This is an aspect that every solution has to handle. The second kind of mobility, “slow individual mobility”, is characterized by a continuous node mobility, where the node movement can be tracked gradually by its neighbors. The third aspect can be defined as “fast individual mobility”. In this case, the node is present continuously in the network but its neighborhood changes completely between two given instants. This mobility does not allow neighbor nodes to track the mobile node movement. The last category of mobility we consider here is group mobility. This type of mobility leads to network splits and mergers, that are the aspects any addressing space (except the geographic one) has difficulty to handle. These different kinds of mobility combined together may create a large quantity of possible scenarios which we can not describe here. In the following we limit our analysis to the individual four categories of mobility.

### 6.1 Slow Mobility

Because of the effect they have on our scheme, we gather the join and leave process with the individual slow mobility of nodes. The join process of a node has been described in the previous section as the construction of the addressing space and the corresponding heuristic. When a node moves or leaves, it produces broken links in the trellis graphs it was present. Because of the introduction of redundancy, the structure remains robust and is able to find alternative paths to route packets that were intended to go through the missing node. Since a trellis is a representation of a Convolutional Encoder, the analogy with error recovery is straightforward. Thanks to a Convolutional Encoder, an erroneous message on a communication channel can be recovered if the number of errors is under the capability threshold of the encoder. We have exactly the same analogy with our trellis structure. While the number of active links remains over the capability threshold of the trellis, we are still able to find a path (obviously longer than the shortest one) to reach an existing destination. When the number of active routes falls under the threshold, the trellis structure does not allow routing and needs to be reconstructed.

The reconstruction of a trellis is a local procedure. This is arguably the best advantage toward mobility, since a trellis reconstruction will not have an impact on the entire network. When the number of routes of a trellis falls under the threshold, the nodes remaining in the trellis restart a bootstrap process as if they were just joining the network. Since their UI remains the same, the protocol is able to guarantee reception of all packets they have to receive, at a cost of some retransmissions.

### 6.2 Fast Mobility

Fast Individual Mobility of a node changes its neighborhood and therefore its topology knowledge changes very fast. In order to be able to handle this kind of behavior, our approach must be able to configure trellis graphs, register addresses and route packets at a higher speed than the node movement. Due to the complexity of the trellis set up, and the influence that could have continuous trellis reconfigurations, we do not claim that our trellis-based VRS is able to handle such node behavior. Nevertheless, we think that our approach will cope with most realistic situations.

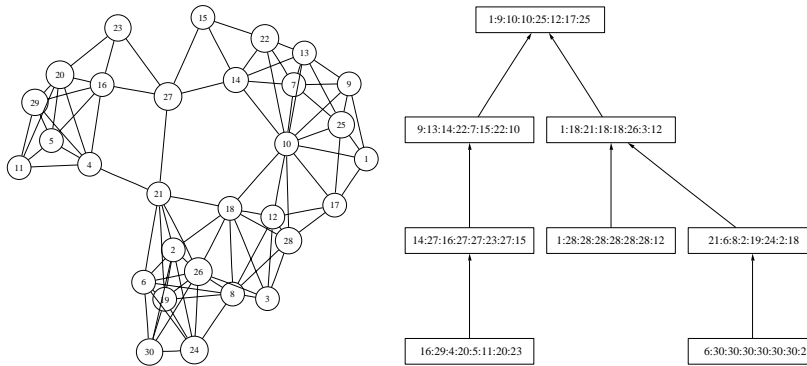


Figure 8: 30 nodes Random Placement Topology and its VRS

### 6.3 Group Mobility

When groups of nodes move in different directions, a network split can occur. Because our VRS is based on a neighbor node construction, the two new-formed network partitions remain coherent, since they form two independent recursive subtrees of the VRS. For each network partition, some nodes representing the missing trellis graphs would have disappeared in different recursive levels. This situation is handled as a successive sequence of node departures. That means that the trellis graphs forming the new border with the other partition may have to be reconfigured. Thanks to our local VRS establishment, the split operation does not influence the entire network, and can be handled locally. If a higher level trellis completely disappears, the prefixes of the remaining trellis graphs have to be updated. This is done by removing the bit string of the new top trellis from all the TP in use.

A network merger can be detected when, for example, two configured nodes  $M$  and  $N$  meet and exchange their highest level trellis Association Table and find no node UI in common. In the ideal case of a network merger, this has no influence on the addressing space, or on the routing topology. In order to exploit this ideal scenario, we bind the two networks thanks to the recursion property of the VRS. In other words, we aim to place one of the VRS of one network “under” the other one. In our example, let say that  $N$ ’s network will remain the “upper” part of the VRS and  $M$ ’s network the “lower” part. The reason for this positioning is that  $M$ ’s VRS contains less recursive levels than  $N$ ’s VRS.

In order to guarantee that the whole VRS remains coherent,  $M$  has to become a part of  $N$ ’s trellis. At the same time, in order to allow packets to be forwarded between  $N$  and  $M$ ’s networks,  $M$  has to be present at the top trellis of its partition. These conditions imply that, when  $N$  and  $M$  meet,  $M$  chooses to join  $N$ ’s trellis because its VRS contains fewer recursive levels of trellis graphs. If  $M$  is a trellis head present at the highest level of hierarchy, it joins  $N$ ’s trellis, and the network merger procedure is finished. If it is not the case,  $M$  starts a trellis head permutation procedure in order to become a trellis head present at the highest level of recursivity.

Our approach takes advantage of the redundant and recursive properties of our trellis-based VRS to handle broken routes, slow nodes mobility and group mobility. Even though we did not describe the mechanisms involved precisely, because of paper length limitation, the advantages of such properties of the addressing space are obvious.

## 7 Evaluation and Simulations

It is always possible to build the VRS containing all the network nodes with recursive trellises. Figure 8 is an example of a resulting trellis-based VRS construction. Since the complexity of such construction is NP-Complete for an existing topology, it is important to simplify the construction as much as possible, as well as to maintain some interesting performance in terms of computation load on the nodes for the routing, minimization of the information storage required for our VRS or matching of the physical minimum route and the path in the VRS to the destination.

Since we use information from the neighborhood to optimize the routing, we expect the trellis-based shortest path to always be in the same order of magnitude as the physical shortest path. We computed the difference of

the path length between the physical shortest path and the path produced by the trellis-based VRS. We do not present the obtained simulation results in this paper, as the number of simulation runs was not sufficient to have a real statistical meaning. So far these first encouraging results give us the indication that the path length does not increase exponentially but they have to be confirmed.

The computation required to route packets along the VRS mostly depends on the length of the path between nodes. The longer the path in the VRS, the more processing time is required to compute it, since more nodes are involved in encoding bit strings representing addresses. Statistically, it seems intuitive that the depth of the VRS will have an impact on the average length of the path between nodes. The depth of the VRS can give us some insight on the performance in terms of average routing processing load for a given topology.

Each trellis stores information on each association (LRA, UID) of its own trellis and of all the trellises of longer prefixes below it. The number of LRA per trellis is fixed, and several LRAs can be associated to one UID. So, in order to optimize the amount of information stored in the VRS, we need to minimize the number of trellises and to create the shortest structure possible. This is clearly what our heuristic tends to do, by limiting the creation of new trellises.

As a result of this analysis, we can deduce that the depth of the virtual structure and the number of trellises has an impact on the performance of our approach. Although our heuristic tends to minimize both metrics, it can not be guaranteed they will both be optimized. One major characteristic of our proposal is that, although we are sure a VRS can be built, we can not predict its resulting shape. As a matter of fact, for a given topology, several VRS can be built, depending on the node arrivals: the number of trellises as well as the maximum number of recursive trellises (depth of the VRS) can differ. It is thus important to understand which parameter will impact performance the most.

In order to evaluate the construction heuristic we proposed, we generated two kinds of topologies. One is a fully-connected topology, where all nodes are 1-hop neighbors. The second one is a randomly placed node topology on a square area of size 1000x1000 meters. Each wireless node possesses a transmission range of 250 meters, a widely used value for simulators (as in NS-2 for example). In these simulations, we studied the construction of the VRS that is the core of our proposal. We tested the heuristic we propose by using a trellis of size  $2^3$ . This parameter remains constant for all of the following simulation results. To evaluate simulation results, we computed the theoretical minimum number of trellis and the minimum depth of the recursive structure one can obtain for a given number of nodes.

## 7.1 Number of trellises

Figure 9 shows the number of trellis graphs produced on the two kinds of topology and the theoretical minimum. On a topology where all nodes are able to communicate in 1 hop (on a clique), our heuristic produces the minimum possible number of trellises. This indicates that the heuristic behaves correctly. This topology is nevertheless an ideal case, as the more nodes are connected, the easier it will be to fill each trellis with new nodes and thus reduce the number of new trellises. For more realistic topologies, where nodes are placed randomly, the upper curve shows that our heuristic remains close to the optimum number of trellises.

## 7.2 Depth of the VRS

Figure 10 shows that the depth of the structure constructed on such a fully-connected topology is very high. This is explained due to the algorithm of our heuristic. Since there is no physical topology constraint in this situation, the first trellis tested for an available position will accept the new node. The other possible trellises are never tested and this leads to a very high depth. We can also see that the curve representing the same situation increases but stays reasonably close. It shows that in a more realistic situation our heuristic behaves pretty well, even with more physical connectivity constraints. Optimizing both the number of trellis and the depth of the structure is really difficult, and can be the next improvement to this heuristic.

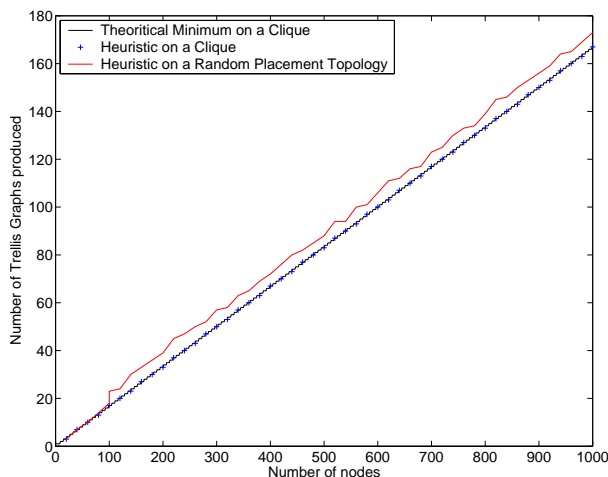


Figure 9: Number of constructed Trellis for different topologies.

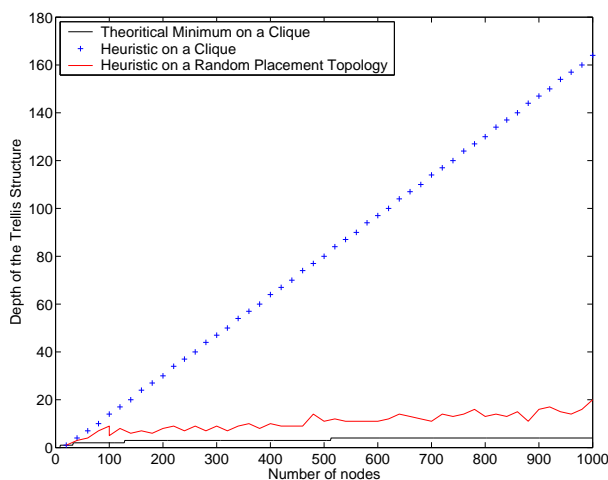


Figure 10: Depth of the Trellis-based VRS for different topologies.

## 8 Conclusions and Further Research

In this paper we have proposed the use of virtual Trellis structures to (a) organize the addressing space of a SON in a structured fashion, and, (b) to perform data routing in such an environment. Trellis structures are well known in classical communication and information theory, but their application in SON is novel. We strongly believe that introducing virtual structures for addressing space that are not based on trees will open new research areas. Thanks to the redundancy and recursion introduced in the trellis-based VRS, we provide a distributed dynamic addressing scheme, with the following advantages: localized operations, no size limit in the addressing space, a built-in multi-path routing structure, computed locally, and robustness to various kinds of mobility. The main disadvantages of our approach are (a) routes are suboptimal, in terms of number of hops, and, (b) control overhead is not fairly distributed among nodes. The construction of the optimal trellises was proven to be an NP-Complete problem; a heuristic was provided.

Future work may include a thorough study of the node position retrieval problem, and a routing performance comparison to approaches tackling similar problems. Since a large quantity of trellis graphs of different sizes, degree and connectivity can be produced, it would also be very interesting to define the criterion to choose a good trellis to produce better VRS.

## References

- [1] M. Mauve, J. Widmer and H. Hartenstein, “A Survey on Position-based Routing in Mobile Ad Hoc Networks,” November 2001.
- [2] J. Li, J. Jannotti, D. De Couto, D. Karger and R. Morris, “A Scalable Location Service for Geographic Ad Hoc Routing,” in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, August 2000, pp. 120–130.
- [3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content-Addressable Network,” in *Proceedings of ACM SIGCOMM San Diego*, 2001.
- [4] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,” in *Proceedings of ACM SIGCOMM San Diego*, 2001, pp. 149–160.
- [5] A.C. Viana, M.D. de Amorim, S. Fdida and J.F. de Rezende, “Indirect Routing Using Distributed Location Information,” in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Dallas-Fort Worth, Texas., March 2003.
- [6] J. Eriksson, M. Faloutsos and S. Krishnamurthy, “Scalable Ad Hoc Routing: The Case for Dynamic Addressing,” in *IEEE INFOCOM 2004, Hong Kong*, March 2004.
- [7] M. Castro, M. Costa and A. Rowstron, “Should we build Gnutella on a structured overlay?,” in *HotNets-II*. Cambridge, MA, USA, November 2003.
- [8] R. Wakikawa, J.T. Malinen, C.E. Perkins, A. Nilsson and A.J. Tuominen, “Global Connectivity for IPv6 Mobile Ad Hoc Networks,” Internet-Draft, November 2002, draft-wakikawa-manet-globalv6-01.txt - Work-in-progress.
- [9] E.M. Belding-Royer, Y. Sun and C.E. Perkins, “Global Connectivity for IPv4 Mobile Ad Hoc Networks,” Internet-Draft, November 2001, draft-royer-manet-globalv4-00.txt - Work-in-progress.
- [10] C.E. Perkins, E.M. Royer and S. Das, “IP Address Autoconfiguration for Ad Hoc Networks,” Internet-Draft, November 2001, draft-ietf-manet-autoconf-01.txt - Work-in-progress.
- [11] J.H. Jeong, H.W. Cha, J.S. Park and H.J. Kim, “Ad-hoc IP Address Autoconfiguration,” Internet-Draft, May 2003, draft-jeong-adhoc-ip-addr-autoconf-00.txt - Work-in-progress.
- [12] D.B. Johnson and D.A. Maltz, “Dynamic Source Routing in Ad Hoc Wireless Networks,” in *Mobile Computing*, vol. 353. Kluwer Academic Publishers, 1996.
- [13] S. Nesargi and R. Prakash, “MANETConf: Configuration of Hosts in a Mobile Ad Hoc Network,” in *IEEE INFOCOM 2002*, New York, NY, June 23-27 2002.
- [14] H. Zhou, L.M. Ni and M.W. Mutka, “Prophet address allocation for large scale MANETs,” in *IEEE INFOCOM 2003*, April 2003.
- [15] C.E. Perkins and E.M. Royer, “Ad-hoc On-Demand Distance Vector Routing,” in *MILCOM '97 panel on Ad Hoc Networks*, November 1997.
- [16] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum and L. Viennot, “Optimized Link State Routing Protocol,” Internet-Draft, March 2003, draft-ietf-manet-olsr-08.txt - Work-in-progress.
- [17] S.D. Nikolopoulos, A. Pitsillides, and D. Tipper, “Addressing network survivability issues by finding the k-best paths through a trellis graph,” in *IEEE INFOCOM, Kobe, Japan*, 1997.

- [18] S. Lin, D.J. Costello, *Error Control Coding: Fundamental and Applications*, Electrical Engineering Series. Prentice-Hall, 1983.
- [19] D. G. Kirkpatrick and P. Hell, "On the Completeness of a generalized Matching Problem," in *Proceedings of the tenth annual ACM symposium on Theory of computing, San Diego, CA*, 1978.