

Self-Stabilizing Network-Layer Auto-Configuration for Mobile Ad Hoc Network Nodes

Timothy K. Forde, Linda E. Doyle and Donal O' Mahony
Centre for Telecommunications Value-Chain Research (CTVR),
Department of Electronic and Electrical Engineering and Department of Computer Science,
Trinity College, University of Dublin, Ireland

Abstract - This paper describes a self-stabilizing protocol which enables the reactive auto-configuration of multi-protocol mobile ad hoc network nodes. The spatially and temporally diverse nature of ad hoc networks suggests the development of a more sophisticated network-layer that enables the network's nodes to choose their preferred routing protocol from an array of protocols with varying abilities. However, the configuration of such a dependent variable, and one with network-wide significance, is a non-trivial task in a distributed system. This paper focuses on the problems that arise in developing a configuration protocol which does not compromise the inherent distributed, open and infrastructure-independent traits of an ad hoc networking system. Self-stabilization is identified in this paper as a suitable technique on which to base such a robust and distributed autonomous configuration protocol. The protocol takes the form of a Stateless Configuration Initialisation (SCI) protocol which is coupled with a Configuration Conflict Detection & Resolution (CCDR) protocol.

Keywords: Auto-configuration, Mobile Ad Hoc Networks, Routing, Self-Stabilization.

I. INTRODUCTION

A mobile wireless ad hoc network is an autonomous collection of routers that have the ability to dynamically and rapidly form networks without the use of any centralised network infrastructure using wireless communication technologies. Ad hoc routing protocols are at the core of ad hoc networks as they allow remote nodes to communicate without resorting to primitive flooding techniques. It is well established in the literature that a one-size-fits-all approach with regard to the choice of optimum routing protocol does not suffice [1], [2], [3], [4]. Rather, in keeping with the distributed and ad hoc nature of these networking systems, it would be preferable to design a networking system that allows nodes to dynamically configure and utilise the most suitable ad hoc routing protocol [5], [6]. Divergent user scenarios and networking conditions, as exemplified by node mobility, node density and traffic loading, demand networking protocols which are tailored to their requirements.

To this end, we propose a flexible network-layer which has access to a suite of distributed ad hoc routing protocols at runtime. In a distributed system such as a mobile ad hoc network, the routing protocol is a dependent feature of the network-layer, i.e. for two adjacent nodes to communicate they must operate the same protocol. Normally, the static configuration of one protocol across the entire network addresses this issue. However, it constrains such a temporally and spatially divergent system. Our introduction of the concept of the ad hoc routing protocol as a variable parameter of the

network-layer necessitates the development of a technique which enables distributed and autonomous nodes to organise this variable, but dependent, parameter which has network-wide scope. To this end, we propose a self-stabilizing network-layer auto-configuration protocol that enables the nodes to collaboratively organise their network-layer routing protocol configuration such that a uniform choice is made across regions of connected nodes.

Section II describes the specific objectives of the auto-configuration protocol and reviews related work. Section III examines a node's ability to observe its environment using ambient signalling. Section IV presents an overview of the key concept of self-stabilization. Section V describes the detail of the self-stabilizing auto-configuration network-layer protocol while Section VI concludes the paper.

II. AUTO-CONFIGURATION

A. Protocol Problem Specification

The aim of the network-layer auto-configuration protocol is simple: each node must operate the same protocol as its neighbours. Accordingly, the state of the network, when viewed as the union of the states of its constituent nodes, must be consistent and stable in the presence of disruptive events.

The specific auto-configuration problems which are considered in this paper revolve around naturally occurring networks events which are experienced during the life-cycle of a mobile ad hoc node, e.g. network mergers, node migration. The auto-configuration protocol must correct the state of the network when incompatibly configured nodes impinge on each other as a result of such network events. Alternatively, nodes may also become incompatibly configured if they proactively reconfigure themselves in response to changed networking conditions, a concept beyond the scope of this paper.

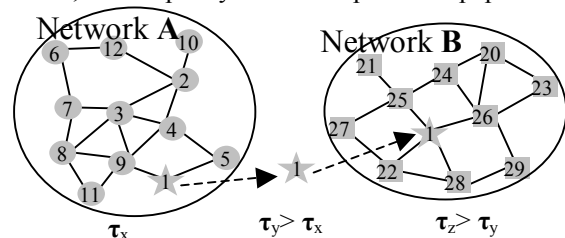


Figure 1. A Node Migrates from One Network to Another.

Consider Node 1 which migrates from one network to another, differently-configured, one, Fig. 1. Node 1 must adapt to the prevailing routing protocol configuration in Network B in a timely manner. In the more complex case, shown in Fig. 2, where two differently-configured networks merge, i.e. where

the two networks come to occupy the same ether, one of the networks should yield to the other network and adapt to the prevailing network-layer technology.

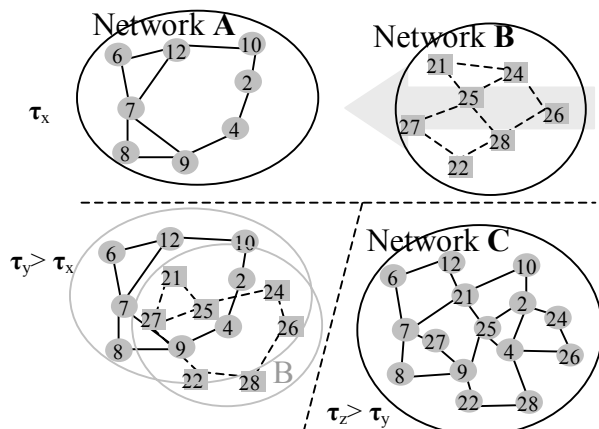


Figure 2. Two Networks Merge

B. Related Work on Auto-configuration

The notion of variable network-wide parameters already exists in the ad hoc networking domain in the form of dynamic node addressing. The concept of router address auto-configuration lends itself to the task of designing a network-layer protocol auto-configuration component [7], [8], [9]. The configuration of unique addresses and the configuration of a uniform routing protocol across a region have orthogonal configuration objectives; the former seeks nodal-configuration heterogeneity, the later seeks nodal-configuration homogeneity. Whilst acknowledging the differences between the two objectives, it is noted that addresses and routing protocols are both dependent and variable parameters of the network. Addresses and routing protocols have non-local significance in a network as collaborating nodes must use both unique addresses and uniform routing protocols to provide end-to-end connectivity.

Analysis of the address auto-configuration literature reveals that there are two primary approaches to the collaborative auto-configuration of addresses in ad hoc networks: consistent stateful approaches or possibly-inconsistent stateless approaches. The stateful techniques involve all nodes having complete knowledge of the address configuration of every node in the network. A node is configured with a new address based on the active cooperation of all other nodes. There is no ambiguity in the network as regards the configuration of any node. On the other hand, the stateless techniques adopt a more localised approach in which global state knowledge is not required. Rather, nodes tentatively configure addresses based on local information and configuration conflicts are resolved as and when they are detected. The literature has shown that signalling overhead involved in the stateful techniques to be high, whilst the configuration inconsistencies incurred by the lower-cost stateless techniques are readily surmountable.

III. A NODE'S PERSPECTIVE USING AMBIENT SIGNALLING

The auto-configuration of a multi-protocol network-layer should not burden the network. The protocol should be sympathetic to the underlying design of the core routing protocols, rather than working against them and detracting from

their desirable traits which have been tailored to their operating environment. We argue that the use of proactive network-layer signalling in the course of using a reactive network-layer routing protocol would be detrimental to the performance of the routing protocol since there is generally a per-packet cost in accessing a wireless medium. A reactive routing protocol throttles back its signalling overhead when demand for its network-layer service falls off. It is therefore desirable that a node should use ambient signalling, i.e. learn from signals that are generated by the core routing protocols, to gather network information rather than originate its own signalling overhead.

The design of a network-layer auto-configuration protocol revolves around a node's ability to make pertinent observations about the environment in which it is operating. A node's ability to make such observations, based on ambient signalling techniques, is characterised by a number of network-layer and physical-layer properties: the signalling activity of the network-layer, the quality of the node-to-node radio links and the connectedness of the node.

- **Active or Inactive Signalling:** actively signalling nodes are nodes that issue packets on a regular basis, i.e. frequently enough to suit a node's routing protocol so that it may discover and maintain connections with neighbouring nodes: the specific minimum intervals depend on the protocols' specifications.

- **Link Quality:** node-to-node links may be characterised by their quality, i.e. whether or not the links are symmetric or asymmetric. The bidirectional link between any pair of nodes may be considered to be composed of two asymmetric (or unidirectional or anti-symmetric) links; one from node u to node v , the other from node v to node u , Fig. 3. Using this link model, it may be said that each node has an incoming link from its neighbour and outgoing link to its neighbour. If the link between the two nodes is asymmetric, then it may be the node's incoming link or its outgoing link that is down. In Fig. 3a, while node u 's outgoing link to v is down, it can receive a signal from node v .

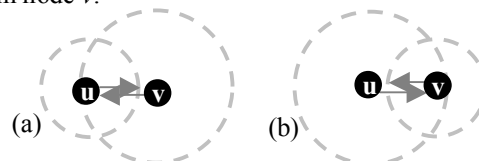


Figure 3. Link Asymmetry may Occur in the Incoming or Outgoing Direction

Such asymmetric links are accommodated in routing protocols such as AODV, DSR and OLSR and should consequently be tolerated in an auto-configuration protocol.

- **Connectedness:** as a consequence of the above-mentioned properties, a node will not always have a full view of its neighbourhood. A node can only know of nodes that it receives messages from. In the case where a node's incoming link is down, the node cannot independently infer whether its outgoing link to another node is up or down. The possibility of asymmetric link quality means that local state knowledge can vary from node to node; in essence it means that the neighbourliness of two nodes is not always mutual. This has an impact on the types of techniques that can be used in designing robust network-layer protocols. We categorise a node as being either isolated from the network, i.e. it has no neighbours, or

connected to the network. A node can establish the fact of its isolation by querying its Route Cache's Neighbourhood Table.

To summarise the effects of these network-layer and physical-layer characteristics on a node's local network perspective, it is helpful to review Fig. 4, which depicts a variety of nodes experiencing the various aforementioned characteristics. In the first case, two groups of nodes occupy the same physical area and ether to the left of Fig. 4. One group of nodes is configured to use AODV (circular nodes), the other DSR (hexagonal nodes). As a consequence of a lack of service demand from higher layers, all of the reactive nodes in this region are silent, leaving them isolated after a time. The absence of signalling in this region means that Node 7 is unaware of the neighbouring presence of Nodes 1, 32, 34, 36, 35, 38 and 33 which are with transmission range of it. While Node 7's immediate possible neighbourhood consists of a mix of both AODV and DSR nodes, the configuration of these nodes has no relevance to Node 7 as it has not attempted to connect to the network. As such, as an inactive node, in the presence of other inactive nodes, there is no conflict between Node 7 and its neighbour's configurations. Problems would only arise if such a node, or nodes neighbouring it, became active in which case the affected nodes would have to deal with a merger event.

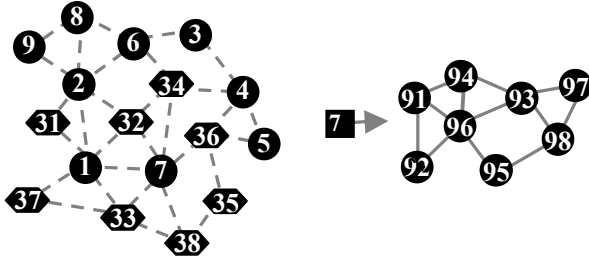


Figure 4. The Effects of Node Isolation and Signalling Activity

In the second case, an active but isolated OLSR (square node) node, Node 29, is moving towards the active region of DSR nodes to the right of Figure 4. As the region of DSR nodes is active, these nodes are connected to each other and are somewhat aware of each others presence, in so far as the node-to-node link quality allows. As Node 29 encroaches on the DSR region both it and the nodes that it impinges upon will detect each other due to the signalling being generated by the routing protocol or packet-switching components of their network-layers. Such signalling, together with both Node 29's and the DSR region's awareness of their connectivity status should enable them to resolve the configuration conflict by means of Node 29 adopting the DSR protocol.

In summary, in the absence of network-layer activity there is no need for a node to actively pursue corrective action. However, when network-layer activity restarts, the information needed to auto-configure the network should be provided by the signalling generated by the core network-layer protocols.

IV. SELF-STABILIZATION

Self-stabilization [10], a concept that was originally pioneered in 1973, when Edsger W. Dijkstra [11], focuses on the ability of a system to converge, within a finite number of steps, from an arbitrary state to a state that exhibits desired system behaviour. In other words, a protocol would be said to

be self-stabilizing if its specification does not require a particular initial state to be imposed on the system to ensure the correct behaviour of the protocol.

Awerbuch *et al.* [12] and George Varghese [13], have refined the concept of self-stabilization by introducing the notions of *local checking*, *local correction* and *counter flushing* among other concepts. In the networking context, the global state of a network is defined as the local state of each node and link, i.e. the messages on the link passing between nodes. Simply and briefly put, the network model is represented in the usual manner by a symmetric, directed graph. Each link in the system may be viewed as an individual subsystem. For every pair of nodes, i.e. each link, there is an arbitrary leader, the selection of which may be a function of the nodes' addresses.

The main thrust of the idea behind local checking is that for many protocols it can be shown that whenever the protocol is in an illegal global state some link subsystem must also be in an illegal state. The efficiency of this approach compared to a centralised global checking one is that state information does not have to be collated at a central node before it is checked for consistency. Each link subsystem can be checked in parallel.

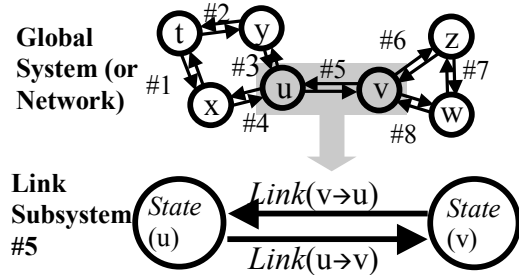


Figure 5 One Global Network, Eight Local Link Subsystems

Correct network behaviour, reflecting the objectives of the network protocol is characterised in the form of a set of legitimate states, or *predicates*. A predicate is simply a property or attribute of the network that can be affirmed or denied by some means, often taking the form of an evaluation function. In practical software coding terms, predicates will often take the form of *if/else* statements that may evaluate conditions (predicates) and control the execution of subsequent protocol steps. A protocol is said to be stabilized when it reaches a global legal state. A network protocol is locally checkable if it has been designed such that whenever the protocol is in a bad state, some link subsystem is also in a bad state and that link subsystem can detect this fact locally. Fig. 5 shows a single networking system of seven nodes which can be viewed as consisting of eight overlapping link subsystems. Link subsystem #5 consists of nodes u and v and the links between them. The local state of the (u, v) link subsystem consists of the 4-tuple of the states of nodes u and v , and the states of the anti-symmetric links connecting them, i.e. u to v and v to u .

A *local predicate* of the global system (or network in this case) is defined in terms of a subset of the local states of that network. Specifically, a local predicate of the network for the (u, v) link subsystem is based on a subset of the states of the (u, v) link subsystem. A local predicate is defined such that only when the local link subsystem of the network, on which it is defined, is in a valid state should the local predicate be satisfied or hold true. A *link predicate* is the term used to describe the collection of local predicates, i.e. one for each link subsystem

in the network. So, for the network depicted in Fig. 5, the link predicate would be the set of the eight local predicates for each of the eight link subsystems. Link predicates are defined such that only when the entire network is in a valid state should the link predicate, as manifested by the subset of states of each of the local link subsystems, be satisfied or hold true.

A. Local checking

The global state of a network protocol is locally checkable using a link predicate set if the global state can be represented by the conjunction of all the local predicates making up the link predicate set. Roughly speaking, the network depicted in Fig. 6 can be seen to be in an illegal state, with regard to the goals of the auto-configuration protocol, as one of its link subsystems is invalid. Link subsystem #1 is in an illegal state as a DSR-configured node is connected to an AODV-configured node. According to the objective of the auto-configuration protocol, neighbouring nodes should operate the same protocol. If the invalid link subsystem was corrected by some means then the entire network would enter a globally legal state. The issues that arise are as to how nodes in the bad link subsystem detect the illegal state and how they correct the link subsystem.

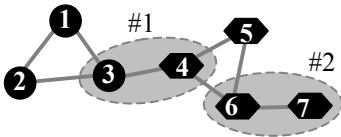


Figure. 6 A Network Featuring An Illegal Link Subsystem

If Node 4 were to attempt to correct the subsystem by configuring itself to use DSR, then link subsystem #1 would be corrected, but the other link subsystems overlapping at Node 4 would be invalidated. To address such a problem, the design of a locally checkable and correctable self-stabilizing protocol demands that the local predicates be *closed* for local checking to be possible. Local predicates are often two-way, i.e. they involve the state of node u , the state of node v , and the state of the links intervening between the two nodes, i.e. the link from u to v and the link from v to u . A local predicate of the network, which is defined by the state or *configuration* of the link subsystem, is closed if the predicate *still holds* when the state of the link subsystem has been subjected to a legal state transition by the network protocol controlling the configuration of the global system. Basically, the protocol must include some rules that detect and exclude data that could send it into an arbitrary state, assuming it is initially in a correct state.

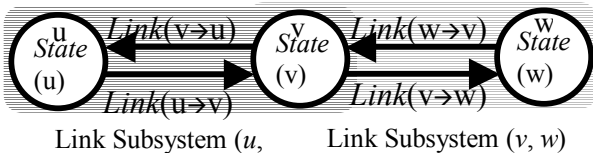


Figure. 7 Two Overlapping Link Subsystems

To clarify the purpose of local predicate closure, consider the system illustrated in Fig. 7 which consists of the two overlapping (u, v) and (v, w) link subsystems. The validity of the global state of this system is equivalent to the conjunction of the two local predicates, $L(u, v)$ and $L(v, w)$. It may occur that one local predicate in this system is always false and that

the false predicate is constantly changing from $L(u, v)$ to $L(v, w)$. As the system is asynchronous, the predicates cannot be checked all the time; they can only be checked on a periodic basis. Therefore, the checking procedure may never detect that there is a fault in the system as the fault continually moves from one link subsystem to the other. This is possible if the protocol can be driven between illegal and legal states by arbitrary state transitions that are not controlled by the protocol. If the network protocol precludes the possibility of corruption, i.e. if the protocol detects bad values and ignores them, then state transition cannot be caused by spurious or corrupt data.

B. Separable Local Predicates and One-way Checking

If the local predicates are chosen such that they are closed, then a snapshot technique can be used to check the state of the local predicate, and thereby to check the global state of the network protocol. Snapshot techniques generally employ a two-way mechanism whereby the leader of the link subsystem, node u , issues a Snapshot Request to the node v , expecting a Snapshot Response in return, Fig 8(a). The response would record pertinent state information regarding the state of node v .

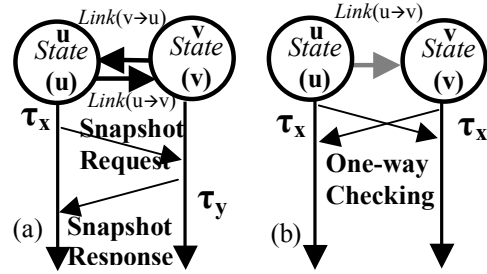


Figure. 8 Two-Way Snapshot and One-Way Checking

However, it has been noted that the typical mobile wireless ad hoc network is susceptible to the occurrence of asymmetric links. This link characteristic renders the use of the two-way snapshot local checking technique unworkable. An alternative and faster one-way snapshot technique has been identified as being suitable under certain imposed conditions [13]. In order to use this technique, the local predicate must be separable in addition to being closed. A separable local predicate is one that may be split into two one-way predicates; one one-way predicate for each link direction. The link predicate for the link subsystem illustrated in Fig. 5, which is dependent on the state of nodes u and v and the state of the two anti-symmetric links intervening between them, is split into two one-way predicates $O(u, v)$ and $O(v, u)$. One-way predicates are designed such that both one-way predicates must independently hold true for the local predicate to hold true. The one-way predicate $O(u, v)$ is checked by node u and is dependent on the state of local variables at node u and the state of node v as described by a message sent from node v to node u over the link. It should be keenly noted that the one-way predicate $O(u, v)$ does *not* involve the state of the local variables at node v or the link from node u to node v . The other one-way predicate for the (u, v) link subsystem, $O(v, u)$, depends on the state of local variables at node v and the state of node u as described by the message sent from node u to node v over the link. In essence, the one-way predicate at node u , for any of its link subsystems, is only dependent on its own state variables and the messages

from the other nodes in its various link subsystems regarding their state.

It has been shown in [13] that if the network protocol is one-way checkable, then it can be locally checked by periodic sending of state, i.e. if nodes u and v periodically send the pertinent state information to each other, then both nodes can check their one-way predicates for violations in lieu of using a two-way snapshot mechanism. As shown in Fig. 8.b, nodes u and v can send state information to each other at time τ_x , rather than checking the link predicate using the slower two-phase approach depicted in Fig. 8.a.

C. Local Correction

A network protocol is locally correctable if an arbitrary (or illegal) global state can be corrected to a desired legitimate global state by applying independent local actions, thus leading to fast and robust stabilization. [13] has proposed the use of a local reset mechanism to restore the local predicate at a link subsystem. Since overlapping subsystems may attempt to correct themselves at the same time, the correction of one subsystem may invalidate the correctness of another overlapping subsystem and may lead to so-called corrective thrashing between the overlapping subsystems. This would occur when subsystems are dependent on each other, and when the dependency is cyclic. Referring back to Fig.7 if the (u, v) subsystem was dependent on the (v, w) subsystem, then the correction of the (v, w) subsystem may cause the (u, v) subsystem to become invalidated. However, if the subsystems can be designed to be independent of each other or if an acyclic dependency can be imposed on them, then the corrective thrashing can be avoided. Referring to Fig. 7, the stability property demands that the application of a local reset function at node u with respect to node v cannot affect the local predicate of the (v, w) link subsystem. An acyclic dependency relation can be assured by imposing a partial order on the link subsystems. If the link subsystem (u, v) is *less* than the (v, w) link subsystem, with respect to the partial order, then the time taken to correct the system is, at most, proportional to the length of the acyclic dependency chain.

V. SELF-STABILIZING NETWORK-LAYER AUTO-CONFIGURATION

A. Introduction

The aim of the auto-configuration protocol is simple; each node should operate the same protocol as its neighbours. As a node only needs to be compatible with its immediate one-hop neighbours, this allows the protocol to be designed around a near-stateless approach which is similar to the stateless address configuration protocols that combine weak global consistency with a local configuration conflict detection algorithm. The scheme is near-stateless, rather than stateless or stateful, as, while it does not require complete knowledge of the global state of the network, it does rely on information about the state of the local neighbourhood.

The design of the auto-configuration component is split into two distinct protocols: the Stateless Configuration Initialisation (SCI) Protocol and the self-stabilising Configuration Conflict Detection and Resolution (CCDR) Protocol. The SCI protocol deals with the configuration of a single node when it boots up. The node's routing protocol is tentatively configured and some

state variables necessary for the operation of the CCDR protocol are configured. Once the SCI protocol has configured those network-layer parameters it terminates and the CCDR protocol becomes operable. The protocols are presented in a number of stages. Local variables and messages that form the basis of the protocols are discussed in subsection B. In subsections C and D, the SCI protocol and the CCDR protocol are presented.

It is also necessary to bear in mind that the auto-configuration protocol is but one enabling element of the multi-protocol network-layer built on the Dublin Ad hoc Wireless Network (DAWN) stack [14]. DAWN is a real ad hoc network testbed that facilitates experimentation with ad hoc networks on all levels from the application layer to the physical layer (e.g. at the security layer, routing layer, MAC layer etc.). At the network-layer of a DAWN communication stack, multiple routing protocols may be run concurrently and protocols can be dynamically started up, closed down and replaced at runtime. The DAWN stack also supports the creation of internetworks of independent, heterogeneously configured, ad hoc regions (or autonomous domains) and the proactive reconfiguration of such regions in response to changed networking conditions. As such, certain region-to-region bridge nodes can simultaneously operate any pair of routing protocols. In describing this protocol, we consider a multi-protocol network-layer which allows a node to choose from AODV, DSR or OLSR.

B. Local State Variables and Node Status Messages

Consider Fig. 9 which depicts a network that is composed of two contiguous regions of nodes. The region on the left uses DSR and the region on the right uses OLSR. Each node-to-node pair of connected nodes forms a link subsystem.

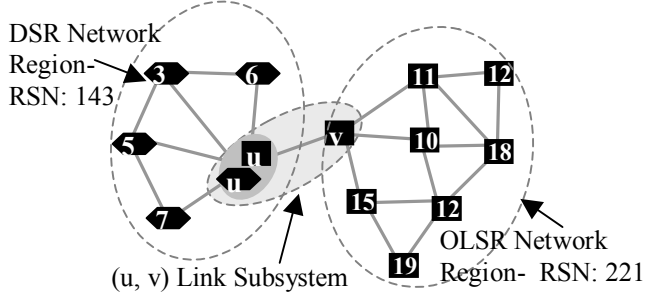


Figure 9 One Network, Two Heterogeneously Configured Regions

The (u, v) link subsystem is fully described by the local states of the two nodes and the messages that intervene between them. Each node's Neighbourhood Table contains additional data fields for each neighbour entry to record the information that is carried in Node Status packets which are issued by each neighbouring node. These local state variables are used in the operation of the protocol. Each entry in a node's Neighbourhood Table is managed in accordance with the associated routing protocol, i.e. each neighbour is entered and removed as dictated by the implicit or explicit route deletion rules for each ad hoc routing protocol.

Active Routing Protocol: This variable simply records the routing protocols that the node is configured to use. The first protocol to be configured in the node is referred to as the **A** protocol and the second one to be configured, if the need arises,

protocols available to it; AODV, DSR or OLSR. As illustrated in Fig. 11, the node then increments its *RSN* from zero to the protocol's prime number value, e.g. a node that picks DSR will now have a *RSN* of 3. In the second step, the node randomly chooses one of the protocol-to-protocol transition open to it.

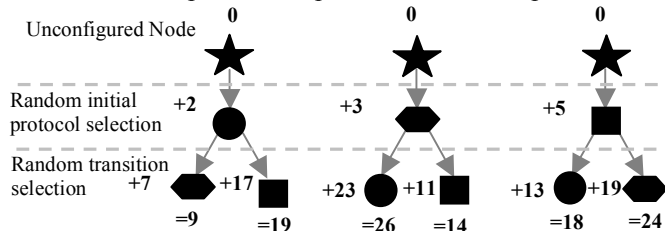


Figure. 11 Two-Round Initial Random Protocol Selection

Again referring to Fig. 11, a node that initially chose DSR can now randomly choose between AODV and OLSR. The node then increments its *RSN* by the protocol-to-protocol transition value. For example, if the node had initially chosen OLSR, and then transitioned to AODV, it would now have a *RSN* of 18. The node now has its A protocol configured, completing the costless stateless initialisation process. The two-round process has the effect of giving each protocol a chance to dominate in the early stages of network creation as each region may be initialised with a high or low *RSN*. As shown in Fig. 11, a DSR node may have a *RSN* of either 9 or 24 and an OLSR node may have a *RSN* of either 14 or 19.

The *RSN* is 1 byte long, i.e. it ranges from 9 to 255, and uses the following wrap-around-proof technique to compare *RSNs*. This technique is used in the OLSR protocol [15]. *RSN1* is said to be "greater than" the *RSN2* if, and only if: $(RSN1 > RSN2 \ \&\& \ (RSN1 - RSN2) \leq (255-9)/2) \ \parallel \ (RSN2 > RSN1) \ \&\& \ (RSN2 - RSN1) > (255-9)/2)$. Thus when comparing two *RSNs*, it is possible, even in the presence of wrap around, to determine the order of the sequence numbers. Also, when a *RSN* cycles through, or to, zero, the protocol-to-protocol transition that takes it past 255 is matched to the nearest initialisation value for that protocol, e.g. if an OLSR to DSR transition wraps around to 12, the number is incremented to 14 so that it aligns with nodes which may have just initialised to OLSR. This process preserves the integrity of the *RSN*. The configured but isolated node is now in a legal state and is ready to network with other nodes.

D. Configuration Conflict Detection and Resolution (CCDR) Protocol

In order to develop the CCDR protocol it is necessary to identify local predicates for the global system that are closed, stable and separable. The auto-configuration link predicates will hold true when the local predicates at each of the link subsystems in the network hold true. The local predicates will be described in terms of *isolated* and *connected* nodes, which have been configured by the SCI protocol, and the Node Status messages that pass between two nodes in each link subsystem. The protocol handles any disruptive events, whether those described in Section II or the purposeful reconfiguration of nodes.

While [13], [14], did not allude to the effects of dynamic network topologies on the use of such techniques, they do show

that a self-stabilizing system can be composed of individually self-stabilizing components. Since the underlying ad hoc routing protocols are inherently self-stabilizing, the effects of the changing topology of an ad hoc network on the CCDR protocol can be mitigated by incorporating the information regarding such changes into the CCDR protocol's design.

In the main, the legal global state only permits homogeneously configured node-to-node links. However, as regional variations in the configuration of nodes are an integral part of the networking system made possible by the flexible multi-protocol network-layer, the auto-configuration component must allow for the existence of intermediary nodes that enable regions to internetwork and co-exist. Consequently, two heterogeneously configured regions may be linked by means of intermediary Boundary nodes operating two protocols simultaneously.

Therefore, the CCDR protocol is based on two separable local predicates. The first predicate checks that a node is configured correctly with regard to the last node from which it has received a Node Status message; this predicate ensures the homogeneity of a region's configuration. The second predicate ensures that a Boundary node, operating two protocols simultaneously, is configured correctly with regard to those nodes for which it is acting as an intermediary.

General Homogeneity Predicate: The first predicate states that Node *u*, the receiving node should operate at least one ad hoc routing protocol which Node *v*, the sending node, operates. It also says that whenever Node *u* and Node *v* do operate the same ad hoc routing protocol(s), that Node *u*'s *RSN* should be greater than, or equal to, Node *v*'s *RSN* for the corresponding ad hoc routing protocol.

Boundary Node Heterogeneity Predicate: The second predicate is based on Node *u*'s Neighbourhood Table entries. It says that Node *u* should only continue to operate two routing protocols if, for both protocols, it has at least one neighbour that is only operating each of those routing protocols. For example, if Node *u* is operating both OLSR and AODV, then there should be at least one neighbour that only operates AODV and at least one neighbour that only operates OLSR.

Both of these predicates are closed; the second predicate is only dependent on local state variables which are controlled by the CCDR protocol itself or the underlying ad hoc routing protocol which adds and removes entries to and from the Neighbourhood Table. Likewise, the first predicate is only dependent on information carried in Node Status messages and the local state variables. The integrity of the Node Status message is guaranteed to be in sequence and unduplicated. This is achieved by means of a counter flushing technique. All corrective action taken by Node *u* is taken locally. Node *u* reacts to the violation of either predicate by changing its own configuration, either its ad hoc routing protocol or its *RSN*, or both. The protocol operates by setting Node *u*'s values to correct the link subsystem rather than attempting to adjust the state of both nodes in the subsystem.

Predicate One: Local Checking and Local Correction: When Node *u* detects that it is not operating the same protocol as Node *v*, it must compare its position in the dependency hierarchy, relative to Node *v*'s reported position, before taking corrective action. Two dependencies are imposed on link subsystems in this protocol. Firstly, all isolated nodes are

equally independent, i.e. each isolated node is a system consisting of one node, and they always yield to *any* other node, whether or not it is isolated or connected. As an isolated node lacks a connection to any other node, there is no reason to attempt to preserve its configuration if it conflicts with another node with which it could network if alternatively configured.

An isolated Node u yields to Node v by taking on its configuration, i.e. adopting Node v 's ad hoc routing protocol configuration and changing its RSN to match Node v 's.

A connected Node u , i.e. a node that is part of a region consisting of at least two nodes, takes corrective action only if its RSN is less than the RSN of Node v . If Node u is operating the same protocol(s) as Node v , but Node v has a higher RSN associated with its protocol(s) then Node u takes local corrective action by updating its lagging RSN to match Node v 's superior declared RSN . This corrective action ensures that a region of nodes that share the same protocol will eventually share the same RSN , enabling each of the nodes to take the same unilateral, but regional, action with regard to that ad hoc routing protocol. If Node u is only operating its A protocol and it receives a Node Status message from Node v indicating that Node v is operating one (or two) different ad hoc routing protocol(s), Node u yields to one of Node v 's ad hoc routing protocols if it has a lower RSN by activating its B protocol and setting it to match the configuration of the corresponding protocol in Node v .

Predicate Two: Local Checking and Local Correction: Node u receives interrupts from the Neighbourhood Table when a neighbour entry is modified, e.g. a neighbour entry is removed or a neighbour that was operating a single protocol starts simultaneously operating two protocols. When Node u is simultaneously operating two protocols itself, it ensures that at least one of its neighbours is operating one of Node u 's two protocols in single operation mode. If Node u does not have at least one such neighbour, it deactivates the first protocol for which it detects this change; the only reason for Node u to operate two protocols simultaneously is to facilitate internetworking between heterogeneous regions.

In making use of the signalling generated by the primary network-layer components, the CDDR protocol is not slow to react to configuration conflicts. By its very nature, a conflict can only occur *when* a node attempts to contact another node. Both isolated nodes and connected nodes detect conflicts in a timely manner. Connected nodes, by definition, must signal regularly or else they lose contact with their neighbours. The ambient signalling provided by connected regions of nodes allows both isolated and connected nodes to detect and resolve any configuration conflicts that may exist. An isolated node in the presence of other isolated nodes can only detect and resolve conflicts if it starts signalling to those nodes and forces them to adopt its configuration. By definition, a node will start signalling if it wants to form a network with other nodes as a result of demands from higher layers. Such an isolated node would, by definition, issue *route requests* or *hello* packets, regardless of the type of ad hoc routing protocol. Such signalling will illicit *route responses* or *hello* packets from the newly connected and configured nodes, which were formerly quiet, isolated and possibly incompatibly configured, thus allowing normal ad hoc networking to occur.

VI. CONCLUSIONS

The diverse nature of mobile ad hoc networks suggests that they should be enabled to choose from a suite of routing protocols at runtime. In making the routing protocol a variable parameter of the network-layer, issues surrounding the collaborative organisation of nodes arise. We have presented a novel robust, very-low cost, reactive auto-configuration protocol, based on the concept of self-stabilization by local checking and correction, which ensures that a consistent, stable configuration persists across connected regions of nodes, even in the event of naturally occurring network events such as node migration and network mergers. Using this stable platform, it may be possible to construct a proactive reconfiguration protocol which adapts the network's configuration in response to the prevailing networking conditions. While this auto-configuration protocol has been developed for use at the network-layer, the design conditions that were imposed on it make it suitable for use at the MAC layer or physical layer.

REFERENCES

- [1] Laurent Viennot, Philippe Jacquet and Thomas Heide Clausen, "Analyzing Control Traffic Overhead versus Mobility and Data Traffic Activity in Mobile Ad-hoc Network Protocols", *Wireless Networks*, July 2004, Volume: 10, Issue: 4, Pages: 447-455, Kluwer Academic Publishers.
- [2] Dmitri D. Perkins, Herman D. Hughes and Charles B. Owen, "Factors Affecting the Performance of Ad Hoc Networks", *Proceeding of the IEEE International Conference on Communications*, 2002. Vol. 4, Pages: 2048-2052.
- [3] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", *In Proceedings of the Fourth Annual IEEE/ACM International Conference on Mobile Computing and Networking (MobiCom'98)*, ACM, Dallas, TX, October 1998. Pages: 85-97.
- [4] Samir R. Das, Charles E. Perkins and Elizabeth M. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks", *IEEE Personal Communications*, Volume: 8 Issue: 1, Feb. 2001, Pages: 16 -28.
- [5] Jeff Boleng, William Navidi and Tracy Camp, "Metrics to Enable Adaptive Protocols for Mobile Ad Hoc Networks", *Proceedings of the International Conference on Wireless Networks*, June 24 - 27, 2002, Pages: 293-298.
- [6] Magnus Frodigh, Stefan Parkvall and Christiaan Roobol, "Future-Generation Wireless Networks", *IEEE Personal Communications*, Volume: 8 Issue: 5, Oct. 2001, Pages: 10 -17.
- [7] Nitin H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks", *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Lausanne, Switzerland, 2002, Pages: 206-216.
- [8] Yuan Sun and Elizabeth M. Belding-Royer, "Dynamic Address Configuration in Mobile Ad hoc Networks", *University of California at Santa Barbara*, U.S.A., Technical Report 2003-11, June 2003.
- [9] Jeff Boleng, "Efficient Network Layer Addressing for Mobile Ad Hoc Networks", *In the Proceedings of the International Conference on Wireless Networks (ICWN '02)*, 2002. Pages: 271-277.
- [10] Shlomi Dolev, "Self-stabilization", *MIT Press*, 2000.
- [11] Edsger W. Dijkstra, "Self-stabilizing Systems in Spite of Distributed Control", *Communications of the ACM*, Nov. 1974, Vol. 17, No. 11, Pages: 643-644.
- [12] Baruch Awerbuch, Boaz Patt-Shamir and George Varghese, "Self-Stabilization By Local Checking and Correction", *Proceedings of the 32nd IEEE Annual Symposium on Foundations of Computer Science*, 1-4 Oct. 1991, Pages: 268 - 277.
- [13] George Varghese, "Self-stabilization by local checking and correction", Ph.D. Thesis MIT/LCS/TR-583, Massachusetts Institute of Technology, 1992.
- [14] Donal O'Mahony and Linda Doyle, "An Adaptable Node Architecture for Future Wireless Networks", in *Mobile Computing: Implementing Pervasive Information and Communication Technologies*, *Kluwer series in Interfaces in OR/CS*, Kluwer Academic Publishers, 2002, pp77-92.
- [15] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, Laurent Viennot, "Optimized Link State Routing Protocol", *IETF MANET Working Group*, December 2002. <http://www.ietf.org>