Mythbusters: Whatever You Thought About MANET Routing, Think Again...

Erik Nordström, Richard Gold and Per Gunningberg
Department of Information Technology, Uppsala University
{erikn|rmg|perg}@it.uu.se

Abstract

Protocol and system designs suffer from myths that may affect their nature and purpose as well as their features and capabilities. We investigate the myths surrounding mobile ad hoc networks (MANETs) and the impact of these myths on the ability to build robust, efficient and simple routing protocols. We find that myths arise because of complexity and ambiguous design and lead to confusing protocol specifications, making the protocols difficult to understand and implement in running systems. For example, both AODV and DSR are confused about their nature as routing protocols and both blur the distinction between routing and forwarding. Building on experiences from implementing AODV and DSR, we dissect these protocols and try to dispel the myths surrounding them. From this dissection, we describe a new routing protocol called Partial Link State routing (PLS). PLS is a synthesis of routing concepts from AODV and DSR with a clearer design description and decoupling of routing and forwarding.

Myth: *A widespread but untrue or erroneous belief or a misrepresentation of the truth.*[1]

1 Introduction

To construct robust, efficient and simple protocols and systems, the concepts forming the foundations on which they are built must be clear, focused and free of myths. Typical myths in protocol design concern the nature and purpose of protocols, features and capabilities. There are a number of recognized myth-prone protocols in computer networking. For example: IPSec, Mobile IP, IP Multicast and IPv6 [9, 11, 22]. We believe myths arise because of complexity and ambiguous design and lead to confusing protocol specifications, making the protocols difficult to understand and implement in running systems.

This paper concerns the myths of mobile ad hoc networks (MANETs) and in particular MANET routing protocols. As researchers having worked with the real-world implementation and evaluation of MANET protocols during the past several years, we have achieved a good understanding of routing concepts in general and how these are applied to MANETs in particular. Having no affiliation to the protocol designers we have implemented two MANET routing protocols: namely AODV and DSR, and experimented with many more. In this implementation process we have experienced

[1] Adapted from the Oxford English Dictionary.

considerable difficulties in dissecting the protocols and we believe that myths are responsible for this.

In this paper we take on the role as mythbusters and attempt to dispel some of the myths surrounding MANET routing protocols. To minimize myth creation and to allow for rapid mythbusting we advocate early prototyping with a closer collaboration between designers and implementors. As a proof of concept to myth-free MANET routing we describe a routing protocol called Partial Link State routing (PLS). PLS builds on the experiences from implementing AODV and DSR. In the PLS design we try to be up-front about the nature of the protocol and we have used early prototyping as an integral part of the the design process. Although PLS can be considered a new protocol, its design is not driven by the urge to drastically improve on current designs in terms of performance or efficiency. Neither does it contain new routing concepts and we believe there is little room for such improvement in this area. Instead, PLS is an exercise in the design, structure and description of systems for MANET routing. The novelty of PLS lies in its clarification of general concepts in MANET routing protocol design and its modular structure, serving to reduce complexity and simplify implementation.

The contribution of this paper is twofold:

1. The mythbusting of MANET routing protocol design and the diagnosis of outstanding issues facing the final progress towards a standard for MANET routing, with a clear focus on system implementation.

2. A design space analysis of MANET routing protocol components and the specification of the PLS protocol based on this analysis, which is a synthesis of our experiences from implementing AODV and DSR.

The paper is structured as follows: In section 2 we give a general diagnosis of why and how myths arise in protocol design. Section 3 gives the necessary background on MANET routing protocols to understand the details of later sections. In section 4 we list some of the myths that we believe exist about MANET routing. Section 5 then gives our view of MANET routing reality through an analysis and structuring of the MANET routing protocol design space. With this background, section 6 introduces the PLS protocol design. Section 7 looks at related work and section 8 concludes the paper.

2 Diagnosis

It is not possible to build simple, robust and efficient routing protocols based on myths. In this section we discuss some of the reasons why protocol designs may suffer from myths and ways to avoid or dispel them.

2.1 Focus and Purpose

If protocol designs are not clear and up-front about the focus and purpose of the protocol it leads to diverging views of it. This raises myths about the requirements to solve a particular problem. For example, there are many myths surrounding the IPv6 protocol and its capabilities compared to IPv4 [11]. The core problem IP is trying to solve is inter-network connectivity and it does that by providing a global address

space. The problem with IPv4 in that sense is its short addresses, which have been remedied in IPv6. However, protocol designers took the opportunity to add other features to IPv6 that are orthogonal to the core problem, such as mobility, security and auto-configuration. The motivation for adding such capabilities to the IPv6 network layer is mostly convenience and efficiency. The trade-off with this approach is that it can also lead to myths that such capabilities cannot be implemented outside of the IPv6 network layer [11]. Hence, our conclusion is that by providing excessive functionality outside of the core solution there is a risk of myths being created that blurs the understanding of the protocols and their original purpose.

However, one problem that exists in protocol design is that the short term benefits of avoiding these issues are not obvious. This is because many of the benefits of myth-free designs are evident only at a later stage of the design process. That is, when a protocol is supposed to be widely-deployed and therefore inter-operability of multi-vendor implementations is a key issue. In these situations, protocols which still contain many myths will be difficult to implement and to inter-operate with as the complexity of protocol specification will cause confusion. One goal of this paper, therefore, is to stress the importance of myth-free designs early on, so that the benefits of such designs can be reaped later on.

2.2 The Complexity Trap

Complexity has caused the quality of the protocol specifications to suffer. The authors of [9] define *"The Complexity Trap"* which we redefine[2] here to be:

The Complexity Trap: A protocol's worst enemy is complexity

Simple protocols are simple to understand. The more complex a protocol is, the less likely we as protocol designers and implementors are to be able to understand it. As a result, a complex protocol is more likely to fail in unexpected ways, the cost of implementing and maintaining it is higher than a simple protocol and it is much harder to evaluate. We propose that any future MANET protocol should be as simple as possible. We believe that this will result in a protocol which is easier to analyze, easier to implement and less likely to contain bugs resulting from complex feature interactions. Complexity of protocols and their implementations have been most analyzed in the field of security where the cost of a security hole is much greater than an error in a routing protocol specification. The authors of [4] describe various real-life cryptosystem failures resulting from over-complexity. We believe that the same methodology and rigor from the field of security should apply to protocol design in general. That is, over-complexity being the key issue in leading to routing protocol errors and that it should be avoided whenever possible. We agree with the authors of [9] that: *"There is no substitute for simplicity"*.

2.3 The Case for Early Prototyping

Early prototyping helps busting potential myths early in a protocol design process. We identify three areas where implementation experience is valuable in the development and standardization of MANET routing protocols and for protocol development in general:

[2] The original quote is *'Security's worst enemy is complexity"*

1. **Complexity** : It is common that drafts grow by revisions, incorporating new features, simply to protect the protocol from criticism that it does not support the features of other protocols. This distracts from the focus of the protocol as argued in section 2.1. The result is at best a protocol which is difficult to implement and at worst an incomprehensible and overly complex protocol specification.

2. **Assumptions** : Many assumptions are made in protocol specifications (e.g., drafts and RFCs) that might not be as applicable in reality as in theory, e.g., how broadcast is used in a MANET network [16]. Some features might also be difficult to implement. One reason for such assumptions is that specification authors seldom are the ones implementing the protocol. Proper implementation-based feedback also does not come until very late in the development process.

3. **Simulations** : Protocol implementations are mainly developed for simulators because it provides a very appealing environment for fast prototyping and testing. Simulators in general provide an approximation of reality and for MANET routing in particular, which deals with wireless radio multihopping, models are very crude. Problems with MANET simulations are discussed in depth in [13]. These problems leads to erroneous results and assumptions regarding protocol robustness to failures and unreliable radio transmissions [16, 3]. Early prototyping can help provide us with a more complete view of the protocol's behavior than exclusively focusing on simulations.

3 Background and Overview of MANET Routing

The key elements of MANETs that make them different from traditional Internet networks are the wireless transmission medium and the mobility. They both put strict requirements on reactivity to topology changes and robustness in the face of routing failures. This has implications on how MANET routing protocols are designed compared to traditional Internet routing protocols. Our focus is on reactive protocols because they differ the most from traditional ones and are therefore more interesting from a design perspective. In the rest of this section we first give a general overview of issues in MANET routing and then continue with a more detailed look at AODV and DSR, which are two of the most well known reactive protocols.

3.1 Routing Approaches

MANET routing protocols are commonly divided into three main classes: *Proactive*, *reactive* and *hybrid* protocols. Hybrid protocols are a mix or combination of proactive and reactive approaches and will not be further discussed here.

Proactive, or table-driven, routing protocols continuously map their surroundings and disseminate this information periodically to all other nodes in the network. Their operation is similar to traditional IP routing protocols, but with optimizations for the MANET environment. Paths through the network are already established when nodes initiate traffic for a destination. This minimizes the delay in communication and allow nodes to quickly determine which nodes are present or reachable in the network. Proactive routing protocols are usually based on either the distance vector (DV) (i.e., Bellman-Ford) or the link state (LS) (i.e., Dijkstra) algorithms. The main difference

between these two is that DV protocols use distributed calculation of the topology while the routing information is propagated in the network, while LS protocols exchange link state among all nodes and then from that derives the network topology from local calculations.

In a resource constrained, highly dynamic MANET, it is not always efficient to periodically disseminate routing information in the network. For this reason, reactive (or on-demand) protocols do not compute complete network topologies, but instead build individual routes to destinations only when there is a need for it, i.e., when nodes initiate traffic to a destination. The reactive protocol typically solicit a route by disseminating a route request into the network, usually by flooding. The destination, or an intermediary node with a route to the destination, sends back a reply with the requested route information. Since routing tables contain soft state, routes time out and are removed when traffic is no longer sent. The overhead grows in relation to mobility and traffic load. The downside of this approach compared to the proactive one, is that each node only has a very limited view of the network. This significantly adds to the delay when the current view is not sufficient. Therefore, it is crucial for nodes to learn as much as possible from the information received.

3.2 MANET Routing Protocol Design

Four routing protocols are targeted for experimental RFC status in the IETF MANET working group [1]. The protocols are the reactive Ad hoc On-demand Distance Vector (AODV) [20] and Dynamic Source Routing (DSR) [12] protocols and the proactive Optimized Link State Routing (OLSR) [8] and Topology Based Reverse Path Forwarding (TBRPF) [18] protocols. However, a recent re-chartering of the working group aims to develop one proactive and one reactive routing protocol standard. This re-chartering we see as an excellent opportunity to take a step back and look at the area with fresh eyes. One concrete result of the re-chartering is the design of the DYMO protocol [7]. We see DYMO as a rewrite of AODV, using different terminology and packet format, but having the same basic functionality. DYMO has removed some features of AODV and made them optional through extensions while also adding some extra features.

3.2.1 Ad hoc On-demand Distance Vector Routing

Ad hoc on-demand distance vector is a reactive routing protocol distinguished by its two phase operation consisting of *route discovery* and *route maintenance*. AODV is hop-by-hop, which means that any node in the network only knows the next hop of the path to a destination. Forwarding information is distributed in the network and is configured in the route discovery phase. Only actively used entries are maintained in the routing table. If there is no routing entry corresponding to the destination of generated data packet, the route discovery phase is initiated. AODV then floods the network with a route request (RREQ). A reverse route state is configured on relaying nodes as the RREQ is broadcasted. If the RREQ is received by the destination or by an intermediate node with a "fresh" enough route to the destination, a route reply (RREP) is sent back by unicast along the reverse route to the originator of the RREQ. Analogous to the RREQ, the RREP configures a forward route state on the intermediate nodes it traverses. A route's "freshness" is decided by sequence numbers that each node maintains for all destinations in their routing table. If newer routing information

is received, as indicated by a higher sequence number, a node updates its information with that in the received packet. The usage of sequence numbers ensures loop freedom.

Once a route is established, AODV goes into the route maintenance phase. It consists of keeping the routing information for its destination targets up to date and to repair routes if they break. If an intermediate node on a path detects a broken link it can do a local repair to hide the fact that the route was ever broken. The purpose is to minimize route down-time and to limit the flooding of control packets in the network. The intermediate node can also choose to inform the source that the route is broken by sending a route error (RERR) to upstream nodes. Link connectivity is monitored through periodic beaconing by sending HELLO messages or using link layer feedback.

3.2.2 Dynamic Source Routing

DSR is an on-demand protocol like AODV, but differs in that it collects information about entire paths to destinations instead of just destinations and a next hop. DSR operates as an extra layer in the network stack, in-between the network and the transport layers. Each packet that is forwarded in the network carries a list of all nodes' IP addresses along a path, from the source to the destination. DSR adds overhead since, unlike AODV, all packets (data and control) need this additional information in excess of the IP header. DSR also has the two mode operation consisting of *route discovery* and *route maintenance*. The route discovery is very similar to AODV's, but the entire path is accumulated in the route request (RREQ) and subsequently sent back in the route reply (RREP), as it travels back to the initiator.

DSR's route maintenance consists of detecting broken routes and finding alternative ones if necessary. DSR uses link layer feedback to detect broken links like AODV. If link layer feedback is not possible it relies on per link acknowledgments instead of periodic beaconing. Unacknowledged data packets are stored in a *maintenance buffer* until they are successfully acknowledged by the receiving next hop. After unacknowledged packets have been retransmitted a number of times, they can be *salvaged*, by using an alternative source route if such exists, or they can be dropped and a route error (RERR) sent back to the source node.

DSR specifies an optional *flow state* extension. The purpose is to reduce the overhead of sending the full source route in every data packet. To achieve this, a hop-by-hop forwarding state based on the source route is established within the network. Packets forwarded with the flow extension only need to carry a flow ID instead of the full source route.

4 Busting the MANET Myths

The MANET routing area is, as so many other areas dealing with protocol design, surrounded by myths that confuse and complicate the understanding of the problems to be solved. Here we try to dispel some of the myths that we find have a particularly negative impact on the understanding of the MANET routing problem and the process of coming up with good solutions.

4.1 Myth 1: Routing Takes Place in the Network Layer

The system for packet delivery in the Internet consists of two components: a *forwarding* system and a *routing* system. The forwarding system deals with taking incoming

packets on one interface, indexing them into a forwarding table based on the information in the packet, and from that deriving the output interface onto which the packet is to be sent. The task of the routing system is to make sure that the view that it has of the network is updated with fresh information. This information is subsequently fed into the indexing table. Forwarding is implemented in the network layer and, although routing is sometimes described as part of the network layer, a routing algorithm is usually an application layer program. The routing and forwarding systems are independent of each other. For example, the IP protocol does not deal with routing and no routing protocol in the Internet deals with how packets are actually forwarded, e.g., using IPv4 source routing (which in fact should be called *source forwarding*) or hop-by-hop forwarding. Therefore, despite a common myth, routing is not part of the network layer as routing and forwarding is separate.

Having an increased need for cross-layer mechanisms and interdependence between the layers, the MANET routing protocol designs (on-demand ones in particular) seem confused about the separation between routing and forwarding. The result of this confusion is less modular designs and harder to implement specifications.

4.2 Myth 2: AODV is a Distance Vector Protocol

AODV is portrayed as a distance vector (DV) routing protocol both by its name and its description in the specification. We argue that this DV focus is misleading and has a negative impact on the intuitive understanding of the protocol. That is because AODV shares few, if any, properties with the the common understanding of how DV protocols function found in the literature [10, 14]. In this understanding, a DV protocol is the distributed routing table computation (e.g., using Bellman-Ford) through local exchange of information between neighbors. This mode of operation is iterative, circular, takes time to converge and gives raise to problems such as looping and count-to-infinity. Since AODV uses explicit request and reply messages, communicating not only with its direct neighbors, but also with end-nodes over multiple hops, it is very different from the traditional DV distributed operation. Hence it does not suffer from the count-to-infinity problem caused by non-converging computation. AODV has explicit error signaling and therefore bad news does not travel slowly like in DV protocols. One question that can be raised in this context is: *What functionality can be removed from a DV algorithm while still being classified as DV?* We believe that AODV, if seen as an evolutionary step from DSDV [21], has clearly passed this threshold. The myth about AODV as a distance vector protocol is confusing and therefore it is harder to evaluate and analyze. We welcome the fact that DYMO, which in turn is an evolutionary step from AODV, has removed the DV legacy.

4.3 Myth 3: DSR is about Source Routing

The prominent and distinguishing feature DSR claims is that forwarding information is carried in each data packet in the form of source routes. A problem with this focus on source routing is that it relates to forwarding and not routing as discussed in myth 1. If the separation between routing and forwarding is to be respected, DSR as a routing protocol should not be defined by a forwarding strategy. The confusion grows as DSR also specifies an alternative forwarding strategy using flow labels. What really should define DSR as a routing protocol is the way it collects information about the network and how it uses that information to build a view of the network topology. How this

view is used in the forwarding process is a separate issue and in a good system design it should be independent from the routing.

We claim that the interesting and important parts of DSR are found in the routing. Unfortunately, the focus on the source route forwarding makes the nature of the routing much harder to grasp. By disassembling the routing part of DSR, we find that the fundamental elements exchanged between DSR nodes are lists of concatenated IP addresses. It uses this information to build a routing cache to store tuples of destination and source route pairs. In essence, the concatenated IP addresses are link state. DSR even specifies the optional usage of a link-cache and Dijkstra's algorithm. To us, this makes DSR closely resemble a link state protocol.

Partial Link State routing (PLS), described in section 6 is in part a response to this confused nature of DSR. PLS has many similarities to DSR, but the focus and description of the protocol is very different as it is up-front about being a link state protocol. We believe that PLS is a better design and description of routing functionality than DSR, although technically they could be very similar.

5 The Reality of MANET Routing

After having examined some of the myths involved with MANET routing, we turn to presenting our view of the reality of MANET routing. We disassemble the various MANET routing protocols into their component parts and describe the trade-offs inherent with various approaches to building each component. It is our belief that attempting to decompose the protocols helps us to better understand each component and its interactions with the rest of the system as a whole. Myths are easier to avoid when a clear view is present of the design choices involved with each component.

We first present our view of how routing protocols in general are structured. We then go on to describe our view of the decomposition of MANET protocols and the design choices which typically are made for each individual component. It is our hope that this section will help the aspiring MANET routing protocol designer to keep his design free of myths. It also prepares the reader for the presentation of PLS in section 6

5.1 Routing Protocol Structure

A routing protocol consists of a number elements in the form of data structures and accompanying protocol logic to maintain them. Figure 1 illustrates their relationships. The central data structure is the Routing Information Base (RIB). The main task of the routing protocol is to maintain the RIB with updated routing information. Routing protocols do this in two ways: 1) adding information to the RIB through *routing information resolution* and 2) removing information through *connectivity monitoring*. If there is a need to constantly have fresh routing information the resolution is performed *proactively* at regular intervals. If the need for information is demand driven, it is done *reactively*, when there is a demand for new information. The connectivity monitoring removes information from the RIB by sensing the environment for events that invalidate any of the resolved information.

The information in the RIB is used to calculate a Forwarding Information Base (FIB), which is a data structure used for forwarding data packets. The FIB is usually a subset of the information in the RIB. The information in the RIB and FIB is usually soft state and specific protocol logic maintains the timeout values of this state during its lifetime.

Routing Element	Design Choice
Routing information resolution	*Destination oriented, path oriented, link state oriented*
Routing Information Base (RIB)	*Next hop table, Path table, Link Cache*
Forwarding Information Base (FIB)	*Next hop table, Path table*
Connectivity monitoring	*Beaconing, Acknowledgments, Link layer feedback*
Forwarding strategy	*Hop-by-hop, source route, flow/path label*

Table 1: Design Space of MANET routing protocols

Table 5.1 lists the design choices for the different routing protocol elements. These choices make up our view of the design space of MANET routing protocols. The remainder of this section reviews AODV and DSR with respect to their respective choices in the design space and discusses pros and cons with the selected approaches.

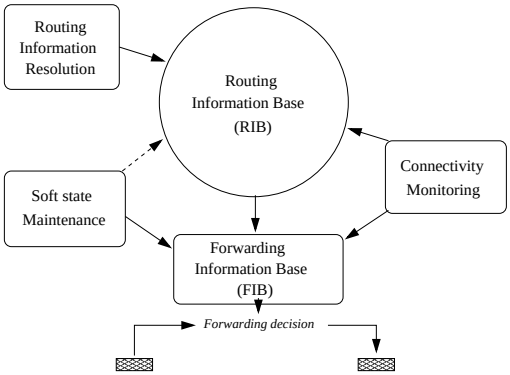


Figure 1: Structure of routing protocols

5.1.1 Routing Information Resolution

AODV and DSR are both reactive protocols and hence routing information is resolved on-demand. Section 3.2.1 and 3.2.2 described their different resolution processes in detail. Although both protocols share many similarities in this regard, they differ in the type of information they resolve and in how it is used. AODV is *destination oriented*, which means that the information retrieved only contains information about one particular destination. DSR, on the other hand, is *path oriented*, meaning the information retrieved is a path or, more specifically, a list of IP addresses of the destination and all intermediate nodes. This approach is also sometimes referred to as *path accumulation* [19]. Path accumulation accumulates the IP addresses of nodes that a flooded request message traverses in the network. Path accumulation has been suggested for AODV and similar functionality is part of DYMO. A third approach is what we call *link state oriented*. The information resolved is then broken down to the fundamental link element that defines the connectivity between two nodes. An example of a link state oriented protocol is OLSR. However, there is an overlap between the path oriented and link state oriented approaches since a path can be seen as a set of link states. Therefore, DSR can be seen as having a link state oriented resolution process if treated as such, although this is not emphasized in the DSR draft.

A reactive link state oriented resolution approach allows nodes to gather information about nodes in the network and their connectivity. This information is, in contrast

to in a “real” link state protocol, gathered reactively and does not necessarily reflect the complete network. The link state oriented approach also easily supports link metrics other than the default hop count. In contrast to AODV style route discovery, where information is aggregated into one metric for the whole path (and hence information is lost), the link state oriented resolution approach is per link. Unlike AODV, where the decision is distributed in the path setup phase, this allows end nodes the flexibility in selecting the best route for their purposes.

5.1.2 Routing Information Base

The routing information base (RIB) is the central storage for the information gathered through the routing information resolution process. This data storage can look very differently depending on the protocol and implementation. The routing algorithm uses the information contained in the RIB to compute, for example, the next hops towards a destination [2]. The computed information is put in the forwarding information base (FIB). Some routing protocols collapse the RIB and the FIB into one data structure. For example, AODV specifies one *next hop table*, for the use as both FIB and RIB, with some extra information such as sequence numbers and “precursors”. In contrast, DSR uses a “route cache” and specifies several ways to implement it, either as a *path table* that consists of tuples of destinations and corresponding path or as a *link cache* as used in link state protocols.

The link cache is clearly the most powerful and generic approach since information is broken down and stored in the basic link element form. Using this information, Dijkstra’s algorithm can be used to compute a network graph. This is similar to link state protocols, except that with reactive resolution of link state the network graph is not computed from all nodes’ link state and only represents a partial network graph. Another advantage is that alternative routes may be computed in case a route breaks. However, one drawback is that information may be outdated, causing extra delay in finding a valid route. This can be tuned by tweaking the timeouts for the state in the link cache.

One argument against using a link cache is the complexity in implementing it. However, in reality it is a quite simple data structure. Considering that the usage of link caches in routing is widespread, the expertise in implementing them is too. The same argument applies to Dijkstra’s algorithm, used to calculate the shortest path using the information in the link cache. The algorithm is well understood and there is a lot of open code available that can be reused. Furthermore, using a link cache in a reactive MANET routing protocol reduces the differences to a proactive link state MANET routing protocol. The result is that more components can be reused (or shared) between fundamentally different routing strategies.

5.1.3 Forwarding Information Base

The forwarding information base (FIB) is the lookup table indexing destinations versus the information needed for forwarding. This information can for example be a next hop or a full path. The FIB is computed from the RIB and hence a subset of the information in the RIB. The purpose is to avoid computation for each packet that is forwarded. Hence, the trade-offs in the FIB design is dependent on the forwarding strategy used and whether the FIB is also the RIB. In that case, the trade-offs are the same as those for the RIB.

5.1.4 Connectivity Monitoring

All ad hoc routing protocols need to monitor the connectivity to their next hop neighbors, to discover either new neighbors or active next hop links that fail due to a node moving out of transmission range. We refer to this process as *connectivity monitoring*, it is also referred to as *neighbor sensing*. Whenever the set of neighbors change, a node might need to react on that change so that data flows transiting that link is rerouted over another path. Connectivity monitoring is maybe the most important feature for how dynamic an ad hoc routing protocol is. The ability to sense the surroundings is what drives the protocol and determines the protocol's quickness in adapting to a changing environment. Therefore, robust and accurate connectivity monitoring is essential for the efficient operation of an ad hoc routing protocol.

Connectivity monitoring is usually part of a protocol's route maintenance and can be done in several ways. The most common way is using *beaconing* by transmitting HELLO messages. This is the method used by, e.g., AODV and OLSR [8]. A HELLO message is a control packet that is broadcasted, announcing a node's presence. They are sent at fixed intervals, for example, once a second. As long as neighbors receive HELLO messages from another node, they will count that node as a neighbor and that connectivity exists between them. If a node has not received a certain number of HELLO messages from another node it will count the link as broken. In the case the link is also part of an active route, the detecting node will take action, either by repairing the route or by sending a route error. HELLO messages add a small overhead due to the periodical control messages broadcasted in the network. An interesting implication of using HELLO messages, particularly in reactive MANET routing, is the proactive component it adds as neighbors are discovered before a route is actually needed. This might in some cases reduce or eliminate route discovery latency. One downside of HELLO messages is that they can cause *Communication gray zones* [16], having a large negative impact on the performance.

A more efficient way to do connectivity monitoring is to use *link layer feedback*. This mechanism is, however, dependent on support in the link layer and a cross layer mechanism to feed the information up the stack to the routing component. To this day, no manufacturer of network cards (802.11 WiFi), has made such detection mechanisms available. Some open source drivers has limited support for link layer feedback, but the functionality is not yet in our experience complete. If link layer feedback is available, the routing protocol does not have to rely on HELLO messages. They can be removed completely, reducing the overhead of the protocol. Link layer feedback is also more accurate and efficient in terms of latency. Link breaks can be detected faster, since the routing protocol does not have to rely on the HELLO messages' timeout, e.g., three seconds in AODV (two lost HELLO messages), to count a link as broken. Link layer feedback is the preferred connectivity monitoring mechanism in simulation. The efficiency and accuracy achieved in simulation portrays the protocol in its best form. However, this also increases the discrepancy with reality, where link layer feedback in general is not available and its is efficiency not proved.

DSR specifies additional connectivity monitoring mechanisms using *acknowledgments*. The first one is passive acknowledgments (pACKs), which uses previously forwarded packets that can be overheard as forwarded again by the next hop. How this is to be implemented efficiently in the network layer is not dealt with in the DSR specification. There are several issues concerning pACKs. First, pACKs only work if a packet is retransmitted by the next hop. This is not the case for one hop paths

or the last hop in a multihop path. The implication is that pACKs can never be the sole mechanism to do connectivity monitoring. This adds to the complexity of the protocol. Second, network layer pACKs require a way to identify packets retransmitted by the next hop and that might prove difficult. There has to be an identifier (ID) element in packet headers that by default can be used by the network layer to identify individual packets. Such a header might have to be added, thus increasing overhead. Furthermore, the transmitting node must maintain a list of unacknowledged packets, or at minimum, an ID counter, also adding complexity. Third, the network interface must operate promiscuously, which adds extra overhead since all packets must be demultiplexed at the network layer instead by dedicated link layer hardware.

A second acknowledgment approach specified by DSR is network layer acknowledgments (nlACKs). An nlACK is a small acknowledgment that is sent back to the previous hop (or piggybacked on other data) upon successful reception of a data packet containing an nlACK request, by a next hop. Since this can add significant overhead, nlACKs do not necessarily have to be sent back for each data packet, but can be delayed for some time, depending on how fast broken links need to be detected. The overhead can still be considerable and interference an issue. This is because forwarding nodes have to queue data packets in the forward direction and nlACKs in the reverse direction at the same time, while being multiplexed over the same network interface. Similarly to pACKs, nlACKs must implement lists or counters of unacknowledged packets. However, nlACKs are more straightforward to implement than pACKs, but also give a larger overhead since, in the worst case, extra packets are sent in two directions at each hop on a path.

An advantage with pACKs and nlACKs is, in comparison to HELLO messages, that they are always transmitted in the same way as data. Therefore, those mechanisms will never suffer from communication gray zones.

An important reality concerning connectivity monitoring is the performance in a real wireless radio environment. Packet reception is not a binary on-off switch according to a distance function between two radio interfaces. Routing protocols are in general event driven and neighbor sensing is the most important event generating mechanisms in MANET routing that enables the protocol to react to mobility. Therefore, it is crucial that a connectivity monitoring mechanism is designed with reality in mind. Unfortunately, this has not always been acknowledged in the MANET community, which to this day, at a too large degree relies on simulation models which are not sufficiently sensitive to the properties of the wireless link.

5.1.5 Forwarding Strategy

Forwarding is the process of taking an incoming packet and through a simple lookup, using the information in the packet, finding the next hop towards the destination. We find three main forwarding strategies in the design space: *Hop-by-hop*, *source route* and *flow/path label*.

Hop-by-hop forwarding is the common routing strategy in IP networks. The distinguishing feature is that a path consists of nodes that only know the next hop towards the destination. This forwarding state is inserted into the network by the routing protocol and enables packets to be forwarded carrying only the final destination. In contrast, source routing does not insert state into the network but instead every packet carries the full path. The obvious downside of this approach is the increased overhead, but it is also more flexible. A salient property of hop-by-hop routing becomes evident when

used in combination with a reactive routing strategy. Reactive protocols maintain a limited routing state of the network at any time, reducing the knowledge of the network to destinations and next hop nodes in active communication. Nothing is known about other nodes in the network or which are part of the paths used. Therefore, a reactive ad hoc network can not easily support the notion of an aggregate route , such as a "default route". This is a problem when connecting two ad hoc networks or when connecting an ad hoc network to one or several Internet gateways [17].

Source routing adds flexibility to the routing protocol at the cost of increased overhead and in some cases complexity. It is easy to support, for example, multipath operation and multiple gateway connectivity. Additionally, many optimizations can be done using the information carried in all packets. For example, DSR can support *automatic route shortening*. A node that overhears a packet being forwarded can notify the sender that one or more hops on the path are unnecessary if it is a subsequent node, but not the current intended next hop, as indicated by the source route of the overheard packet. DSR is intuitively loop free, since loops can easily be detected by inspecting the source route of data packets. Therefore, DSR does not have to rely on sequence numbers as AODV does.

5.2 Evaluation of Design Choices

After having discussed the design choices and their respective trade-offs, we evaluate the implications of those choices in actual routing protocols. The purpose of this evaluation is not to compare performance but rather to highlight differences in protocol logic that stem from the design choices. From earlier experimental evaluation of MANET protocols we have found a scenario we call *Roaming node* to be well suited for highlighting differences in protocol designs. We have recreated this scenario in the ns-2 simulator, based on the original scenario created for our real-world experimental testbed [15, 16]. A simulator fi ts our purpose well because we want to study protocol logic rather than the effects of, e.g., a wireless transmission medium. A constructed, simple and well understood scenario also serves our purpose better than comparing results obtained from a number of random simulation runs, because then it is very hard to understand the connection between cause and effect.

5.3 Experimental Setup

Figure 2 illustrates the Roaming node scenario that we have implemented for the ns-2 simulator version 2.28 running on Ubuntu Linux version 5.10. Three stationary nodes (labeled 0, 1 and 2) are placed in line at positions A, B and C respectively. They have connectivity only with their adjacent neighbors. A fourth node (labeled 3) is mobile according to the dotted line, moving from the start at position A, to position D and then back to A at a constant speed of 1.3 m/s. During this time, node 3 sends a total of 2480 Ping echo requests to node 0 at a rate of 20 packets/s (i.e., for 124 seconds total). Node 0 replies with an echo reply whenever a request is received. The traffi c used is therefore a two-way constant bit rate (CBR) pattern. The CBR flow samples the connectivity between nodes as the scenario plays out. It allows us to measure the round trip time and route taken for packets in both the forward and reverse direction. The simulation runs for a total of 150 seconds with a bootstrap phase of 20 seconds.

To achieve the connectivity patterns necessary for the Roaming node scenario we altered the parameters of the wireless PHY model in ns-2 to better match that of real

11 Mbit 802.11 cards. The wireless transmission range is set to 33 meters using the "TwoRayGround" propagation model to approximate the range in our experimental indoor environment. We have validated that this gives connectivity changes between node 3 and the stationary nodes close to the times of that in real life (within the range of seconds).

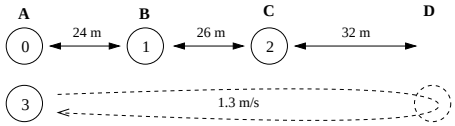


Figure 2: Overview of the Roaming node scenario.

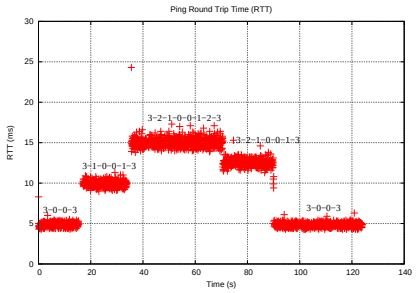
We use our own AODV-UU and DSR-UU protocol implementations in the evaluation. AODV-UU is version 0.9.1 and DSR-UU is version 0.1. Both use the default configuration settings unless otherwise specified. AODV-UU and DSR-UU were each run twice using different connectivity monitoring: either link layer feedback or the best available alternative. In this case, HELLO messages for AODV-UU and network layer ACKs for DSR-UU. The reason for varying the connectivity monitoring is that the mechanism used, as mentioned previously, drives a reactive protocol and has a big impact on its responsiveness. Furthermore, as we shall see, it has implications on the general behaviors of the protocols.

5.4 Analysis

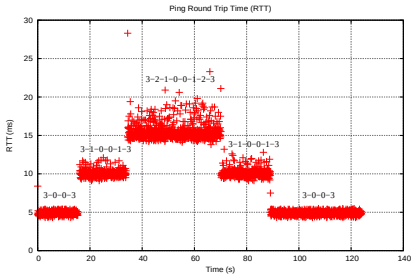
Figure 3 shows, for each simulation run, the round trip time (RTT) for each successfully received reply packet at node 3, versus the time the corresponding echo packet was sent. The prevalent observation is the stair case effect that corresponds to the differences in RTTs between the packets received over different multihop paths. The actual paths taken in forward and reverse directions are printed out in the graphs. Comparing the graphs we can see behavioral differences that are characteristic for each protocol.

DSR-UU has a large variance in RTT when using network layer acknowledgments (nlACKs). The reason is the increased link contention caused by the nlACKs. The variance is similar to AODV-UU with HELLO messages for one hop because nlACKs are few and sent only over one link. The RTT variance increases with the hop count due to the increased number of adjacent and hence contending links that are monitored using nlACKs. With link layer feedback enabled in both protocols, DSR-UU has similar variance to AODV-UU, thus providing confidence in that the nlACKs are the cause of RTT variance. We have noted in other experiments that DSR-UU's variance in RTT has an effect on TCP. A higher variance means a longer TCP timeout, leading to reduced performance.

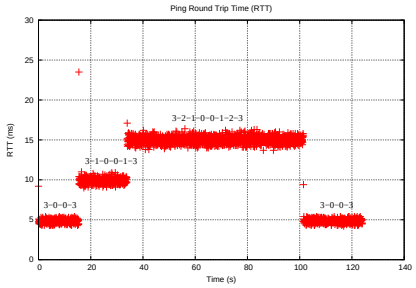
There is a pronounced difference in AODV-UU's operation when using HELLO messages compared to using link layer feedback. HELLO messages work proactively by optimizing the route on the way back from position D, i.e., the reception of a HELLO message will shorten the route. When node 1 receives a HELLO message from the roaming node 3 on the way back (time 70 s), it will optimize its route to the node 3 from two hops to one hop. This is because node 3 is the IP destination of the packets in the reverse direction. An interesting point here is that node 0 will have an erroneous view of the network because it still has a hop count of three to node 3 in its



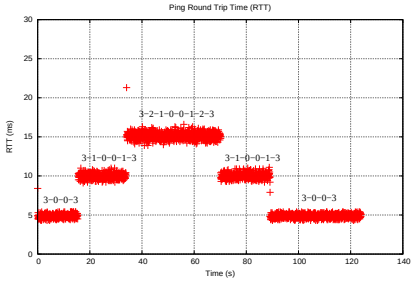
(a) AODV-UU with HELLO messages. Notice a slightly higher variance in RTT compared to using link layer feedback. However, this is not as prevalent as in the DSR-UU case. HELLO messages work proactively and optimize the route on the way back, but not as effi ciently as DSR-UU.



(b) DSR-UU with network layer acknowledgments (nlACKs). Notice the increase in RTT variance caused by the rising link contention from the per link nlACKs as the hop count increases.



(c) AODV-UU with link layer feedback. Without the addition of proactive HELLO messages, AODV-UU is not triggered to seek shorter routes until there is connectivity loss on the longer route.



(d) DSR-UU with link layer feedback. Without the contention caused by nlACKs the variance in RTT increase much less when the route becomes longer.

Figure 3: PING Round Trip Times (RTT) for AODV-UU and DSR-UU in the Roaming Node scenario. DSR-UU always use the shortest route due to its more complete view of the network compared to AODV-UU. The higher RTTs of some few packets are due to buffering during route discovery.

routing table. This is because the route optimization happened between node 1 and 3 which is a sub-path of the path between node 0 and 3. Since the packets in the forward direction from the node 3 have a destination IP of node 0, node 3 will not optimize its route until it receives a HELLO from node 0 at time 90 s. The route is therefore asymmetric (3-2-1-0-0-1-3) between time 70-90 s.

With link layer feedback AODV cannot reap the benefi ts of the proactive HELLO messages. Therefore, AODV-UU will not optimize the route until there is a link break between the roaming node and node D at time 100 and hence the hop count is much longer with link layer feedback. This has an effect on TCP throughput as it degrades with the number of hops.

DSR-UU has a better view of the connectivity in the network since it is path ori-

ented. It can use its knowledge about the entire paths to optimize the hop count for the flows. In DSR this feature is called *automatic route shortening* and allows DSR to always achieve an optimal hop count.

5.4.1 Conclusions

Although our evaluation is simple and in no way covers all aspects of the protocols' behaviors, it allows us to make a number of interesting observations. From these we conclude that the different design choices in AODV and DSR has an non-negligible effect on the their operation. AODV's limited horizon when it comes to the view of the network has an effect on its ability to react on changes in the network topology. We noted two issues: 1) the inability to achieve shortest path in some circumstances and 2) end nodes having invalid/erroneous hop counts for some routes. The impact and importance of these issues on performance can be debated and needs further investigation. DSR, on the other hand, due to dealing with full paths has better view of the network. The actual path taken will always match the intended path.

Another conclusion is that the approach to connectivity monitoring has a significant impact on the performance and reactivity of the protocol in the face of link failures and topology changes. Expectedly, link layer feedback is the most effi cient mechanism, but it is seldom available for use in the real world. Network layer acknowledgments have issues with contention over adjacent links. This effect might be tunable with more effi cient implementations. Hello messages have a proactive element that nlACKs do not have, mitigating some of the poor route optimizations in AODV when enabled. On the other hand, nlACKs do not suffer from Communication gray zones [16], which HELLO messages do. However, this fact is not evident from the simulations described here.

Our fi nal and general conclusion is that there are many desirable properties inherent from the better view of the network that DSR has. This has been one motivation for the design choices in PLS, which we describe in the following section.

6 Partial Link State Routing (PLS)

Partial Link State routing (PLS) is an exercise into the design and description of MANET routing protocols. The motivation behind PLS is to use our implementation experience to inform the creation of a new protocol that should aim to be as free of myths as possible. We then pursue a course of early prototyping to bust any remaining myths as quickly as possible. Our approach consists of:

1. Having a clear description of the nature of the protocol.
2. Keeping routing and forwarding clearly separated. This should be reflected both in the design specifi cation and in the implementation.
3. Refrain from multiple choice design options which can potentially lead to overcomplexity. When there are several options (e.g., forwarding strategy) chose one mandatory default but provide an infrastructure to support other.

PLS is a multihop routing protocol optimized for constraint networks, such as ad hoc and sensor networks. Distinguishing for PLS is the explicit use of link state as the means of building forwarding paths and the gathering of link state reactively. In this

regard, PLS shares similarities with DSR. DSR specifies the usage of a route cache dealing with full paths, while the explicit usage of link state is optional. One goal with PLS is its focus on being a link state protocol. Therefore, with PLS we separate the task of gathering and disseminating routing information from that of forwarding.

A guiding principle of PLS is to keep information at the finest level of granularity and not to aggregate or discard information unnecessarily. Link state (LS) is therefore the fundamental element exchanged in PLS networks as it is the smallest atomic piece of information available about a network. The reactive gathering of link state has the implication that PLS, in contrast to established link state protocols, does not periodically receive link state updates from all nodes in the network. Instead PLS probes the network for link state information when needed. This means that PLS nodes usually only have a partial view of the connectivity in the network and this view varies from node to node. However, it is rare that a proactive protocol actually uses all of the paths potentially available to it since a node typically communicates only with a subset of available nodes. Usually, many nodes use the same paths in the network, for example because it provides high throughput or it leads to an Internet gateway. Therefore, a full link state view of the network is not always worthwhile. The goal of PLS is instead to make sure that a node's view reflects the connectivity information necessary to maintain its communication with other nodes, and that the view is valid.

6.1 PLS Structure

PLS consists of two major components, the Link State Resolution Engine (LSRE) and the Link Cache (LC). In short, the LSRE is what defines PLS and contains logic to make sure that a node has the required link state to compute a path from itself to a particular destination (in PLS terms called *target*). The link cache is the central storage for the gathered link state and represents a nodes current (and partial) view of the network topology.

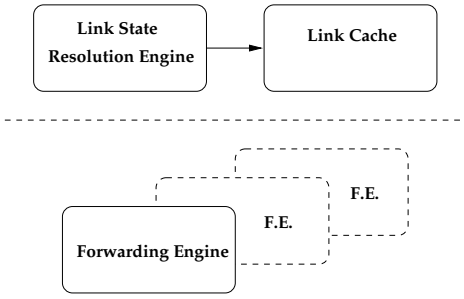


Figure 4: Partial Link State (PLS) protocol overview showing the two main PLS components (LSRE and LC) and multiple possible FEs.

As a link state approach allows great flexibility in the forwarding strategy used, PLS can support multiple forwarding strategies. For this purpose PLS defines a Forwarding Engine (FE) component. The purpose of the FE is to define the required functionality and API for the rest of PLS to function efficiently together with the FE. However, the actual forwarding strategy used, e.g., hop-by-hop, source route forwarding or path/flow label is out of the scope of PLS. The exact requirements on the minimal functionality of the FE and the API used is something we are currently investigating.

PLS control messages deal exclusively with link state. PLS link state is a tuple in

the form:

$$< IP_{src}, IP_{dst}, Cost, Timeout > \quad (1)$$

and is carried in PLS control messages. Nodes can optionally listen to PLS messages promiscuously to pick up extra link state that might prove useful later and hence reduce the amount of link state probes. The following sections give an overview of the different components of PLS.

6.2 The Link State Resolution Engine

The Link State Resolution Engine (LSRE) is a link state oriented routing information element as defined in section 5. The main purpose of the LSRE is to add and remove link state to the link cache in PLS nodes. Whenever a PLS message is received by an PLS node, it passes the message to the LSRE. The LSRE examines the message and adds any link state carried within to the link cache.

When an PLS node wants to send a packet, but no path is currently configured for the target, it consults its link cache. If a path can be computed from the information in the link cache, the node invokes the FE to forward the packet. If no path can be computed, the PLS node invokes the LSRE to resolve link state that can be used to compute a path.

6.2.1 Link State Resolution

To resolve link state, LSRE constructs a Link State Resolution Query (LSRQ) for the destination in question. The LSRQ is disseminated into the PLS network using network layer broadcast. Each LSRQ contains a propagation counter. This counter is incremented by one each time the message is propagated by a node. The node that disseminates a LSRQ is referred to as the *originator* of the LSRQ and the sought destination the *target*. As the LSRQ propagates in the network, nodes record the incoming link in the newly received LSRQ in link state form. The LSRQ should also indicate in which order this link state was added to the message so that it is possible to reconstruct the path the LSRQ has traveled from the information in the LSRQ only.

Each node that receives a LSRQ first adds any link state carried in the LSRQ to its cache and then consults the link cache for a path to the target. If a path can be computed, it appends the missing link state to the LSRQ, sets a *resolved* flag in the message and disseminates the message into the network using a network layer broadcast. The resolved flag indicates that the LSRQ has been resolved and should therefore only be propagated to nodes part of the recorded path in the LSRQ. I.e., any node receiving a LSRQ with the resolved flag set and is not in the path that can be reconstructed from the message should drop it silently. Hence, a resolved LSRQ is sent back using application layer hop-by-hop forwarding, allowing nodes overhearing the message to add the link state carried within in their link caches. Any node that propagates a LSRQ with the resolved flag set should decrement the propagation counter by one. The propagation counter functions as an index into the path carried in the LSRQ, indicating the current node. If a node receives an unresolved LSRQ, and can not itself resolve it, it records the incoming link in the LSRQ, increments the propagation counter and disseminates the message again.

We are currently working out the exact details of the LSRE through the use of early prototyping. We plan to investigate a range of approaches that aim to gather as much link state as possible at the least possible cost.

6.2.2 Dampening

LSRQ messages that are propagated in the network should use dampening to avoid unnecessary propagation of the message that could strain the network resources. A LSRQ that is unresolved is never retransmitted by nodes that are already in the path carried in the LSRQ. A LSRQ identifier carried in each LSRQ, assures in combination with the target and originator IP addresses that the an unresolved LSRQ message is never broadcasted more than once by each node receiving it.

6.2.3 Invalidated Link State Notification

Since PLS have no periodic updates of link state it has to rely on explicit notification of invalidated link state. Such notifications are sent in the form of Invalidated Link State (ILS) messages. When a node invalidates link state by discovering a broken link through connectivity monitoring, it sends an ILS message to all nodes that are part of a path that is affected. To be able to know about these nodes, each PLS node must maintain a list of all paths in active communication of which the node itself is part of.

6.3 Link Cache

The link cache (LC) functions as a RIB and is the central data storage for routing information. The link cache should support multiple routing metrics and allow nodes to maintain a partial view of the connectivity in the network. The view is dependent on what link state the node in question has gathered and is assumed to be different for all nodes. The link state should have a “best before” date, and therefore should time out when unused for a certain period of time. This timeout should be tuned to match the mobility in the network. PLS uses Dijkstra’s algorithm to compute shortest paths to other nodes in the network for which there is link state in the LC.

6.4 Forwarding Engine

The forwarding engine (FE) should implement a FIB suitable for the forwarding strategy used. The FE should export a standardized API for PLS to manipulate the FIB that is flexible enough to support several forwarding strategies and FIB configurations.

6.5 Connectivity Monitoring

PLS needs connectivity monitoring to discover broken links. This is preferably done through link layer feedback. However, any approach, such as beaconing or using acknowledgments is suitable.

6.6 Outlook

We are currently in the process of implementing PLS in ns-2 and the Linux operating system and we expect the first version to be available shortly. PLS is therefore work in progress but we believe that the details of PLS described in this section still provides

a valuable insight into the design philosophy of PLS, which is a result of the analysis throughout this paper.

7 Related Work

In [5], the authors describe an approach to decomposing MANET routing protocols into parameterized components which can be reused between different protocol implementations. The goal of this work is provide higher-quality protocol implementations through component reuse. The work presented in this paper advocates the untangling of the various components of the *design* of MANET routing protocols. We believe that the MANET working group has effectively blurred the boundaries that used to exist in the layered architecture of the TCP/IP protocol stack, and that we need to return to a more decoupled architecture in order to prevent myths from occurring.

The authors of [9] have performed a cryptographic evaluation of the IPsec security protocol. They found the protocol to be overly-complex and overly-flexible thus making it difficult to analyze, implement and evaluate. Their analysis of the protocol matches well with our evaluation of the MANET routing protocols. Namely that complexity is the worst enemy of a (security) protocol. The authors themselves say: *"additional complexity and bloat is seriously detrimental to a normal (functional) standard."*. We believe that simplicity should be brought into the protocol design process as a critical virtue to prevent the problems described by the authors of [9] occurring to any protocol.

We note here that most discussion on the topic of myths or excessive complexity in protocols takes place in presentations such as [6, 22, 11] or other informal arenas. Whilst it is healthy for these discussions to take place in these arenas, we believe that these issues are important enough to be discussed in research papers which are properly archived and distributed by the community.

8 Conclusions

This paper has presented the concept of myths in the context of protocol design. It is our belief that myths hinder the development of simple, efficient and robust protocols by making either the goals or the mechanism of a protocol unclear. This lack of clarity leads to an increase in the complexity of the protocol specification as the true nature of the protocol itself is obscured. We believe that the worst enemy of good protocol design is complexity (i.e., "The Complexity Trap") and therefore it should be avoided at all costs.

Through our implementation experience of two MANET routing protocols, namely AODV and DSR, we have identified a number of myths that are present in the design and specification of these protocols. We note that early prototyping of these protocols could have caught some of these myths, but unfortunately most implementation effort early on in the design of these protocols was almost exclusively focused on simulations which do not provide enough realism for all mythbusting. We have attempted to bust these myths by using our implementation experience to get at the core nature of these two protocols. We have shown how key protocol design decisions lead to differences in behavior, however the performance effect of these behavioral differences is unclear.

Based on our understanding of protocol design decisions, we have created a routing protocol called Partial Link State (PLS) routing. The motivation behind this protocol

is an attempt to create a myth-free protocol by being as clear as possible from the beginning as to the true nature of the protocol. Also, we have attempted to make the protocol as simple as possible, thus avoiding "The Complexity Trap".

We believe that the discussion of myths and their corresponding effect on protocol complexity is an important topic for all aspects of protocol design and implementation. Additionally, we also believe that early prototyping in as realistic a setting as is feasible should be a fundamental part of any protocol design process in order to minimize the effects of myths and to bust them as early as possible.

References

[1] The official IETF MANET working group webpage. `http://www.ietf.org/html.charters/manet-charter.html`.

[2] *Open Systems Networking: TCP/IP and OSI*. Addison-Wesley Publishers, 1993.

[3] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of ACM SIGCOMM 2004*, pages 121–131, August 2004.

[4] R. Anderson. Why Cryptosystems Fail. In *Proceedings of the 1st ACM conference on Computer and communications security*, 1993.

[5] F. Bai, N. Sadagopan, and A. Helmy. BRICS: A building-block approach for analyzing routing protocols in ad hoc networks - a case study of reactive routing protocols. In *IEEE International Conference on Communications (ICC)*, June 2004.

[6] S. M. Bellovin. Lessons from ipv6. Next-Generation Secure Internet workshop, 2005. `http://www.cs.columbia.edu/~smb/talks/ipv6-lessons.pdf`.

[7] I. Chakeres, E. Belding-Royer, and C. Perkins. Dyamic MANET O-demand (DYMO) routing, June 21 2005. Internet Draft, draft-ietf-manet-dymo-02, work in progress.

[8] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, and a. Qayyum et L. Viennot. Optimized link state routing protocol. In *IEEE National Multi-Topic Conference (INMIC 2001), December 2001*.

[9] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec. `http://www.schneier.com/paper-ipsec.pdf`.

[10] C. Huitema. *Routing in the Internet*. Prentice Hall, 2nd edition, 2000.

[11] G. Huston. The Mythology of IP Version 6. *The Internet Protocol Journal*, 6(2):23–29, June 2003.

[12] D. B. Johnson, D. A. Maltz, and Y. Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR), April 2003. IETF Internet Draft, draft-ietf-manet-dsr-09.txt, (work in progress).

[13] S. Kurkowski, T. Camp, and M. Coagrosso. MANET simulation studies: the Incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(4), October 2005.

[14] J. F. Kurose and K. W. Ross. *Computer Networking*. Addison Wesley, 3rd edition, 2004.

[15] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin. A large-scale testbed for reproducible ad hoc protocol evaluations. In *Proceedings of IEEE Wireless Communications and Networking Conference 2002 (WCNC'02), March 2002.*

[16] H. Lundgren, E. Nordström, and C. Tschudin. Coping with communication gray zones in IEEE 802.11b based ad hoc networks. In *Proceedings of The Fifth ACM International Workshop On Wireless Mobile Multimedia (WoWMoM)*, September 2002.

[17] E. Nordström, P. Gunningberg, and C. Tschudin. Comparison of forwarding strategies in internet connected manets. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 8(4):72–76, October 2004.

[18] R. Ogier, M. Lewis, F. Tempelin, and B. Bellur. Topology broadcast based on reverse-path forwarding (TBRPF).

[19] C. Perkins, E. Belding-Royer, and I. D. Chakeres. Ad hoc on-demand distance vector (aodv) routing, July 2004. draft-perkins-manet-rfc3561bis-01.txt.

[20] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing, July 2003. IETF Internet RFC 3561.

[21] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the conference on Communications architectures, protocols and applications SIGCOMM '94*, volume 24, October 1994.

[22] R. Perlman. Myths, missteps, and folklore in protocol design. Invited talk, USENIX'01, 2001. `http://www.usenix.org/publications/library/proceedings/usenix01/invited%talks/perlman.pdf`.