UNIVERSITY OF CALIFORNIA

SANTA CRUZ

SANTA CRUZ ROUTING INFORMATION PROTOCOL

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Spencer Neuschmid

June 2020

The Thesis of Spencer Neuschmid
is approved:

_____
Professor J.J. Garcia-Luna-Aceves, Chair

_____
Professor Katia Obraczka

_____
Professor Brad Smith

_____
Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

ABSTRACT

Santa Cruz Routing Information Protocol

Spencer Neuschmid

The Santa Cruz Routing Information Protocol (SCRIP) is a new routing protocol designed to solve performance problems present in current routing protocols used in the Internet. The routing protocols used in the Internet today were created over twenty five years ago and were created with strict limitations of storage and bandwidth in mind. SCRIP builds on the foundation of the RIP to provide shortest path routing for networks within autonomous systems. The problems that SCRIP solves include routing loops, high storage and signaling overhead, and convergence times that may become too long. These problems are solved by maintaining reference distances, supporting both on-demand and proactive routing, and implementing other techniques for the efficient exchange of distance-vector information. The main idea that allows SCRIP to be loop free and therefore more efficient, was introduced in Ordered Distance Vector Routing (ODVR). ODVR is a routing protocol used in wireless ad-hoc networks that showed that it was possible to maintain loop freedom through distance values alone. A formal proof of correctness and completeness shows that SCRIP is able to exhibit loop freedom at every point in its operation. Simulation experiments using ns-3 show that SCRIP performs better than RIP and OSPF in terms of convergence and signaling overhead in a variety of scenarios.

Acknowledgements

I would like to start by thanking Professor JJ Garcia-Luna-Aceves for overseeing this work and providing guidance and inspiration to pursue such a challenge in this field. Professor Garcia-Luna-Aceves has been an integral part to the main ideas introduced in this thesis, along with teaching me invaluable lessons in researching, networking, and proof writing. I would like to thank Professor Katia Obraczka for first introducing me to the field of computer networking many years ago and teaching me the fundamentals of the work and ideas in this field. She has always given great advice and been very insightful when I have come to her with questions. I would like to thank Professor Brad Smith for believing in this idea and supporting this work before ever knowing me, when the project was still in the beginning phases. I would also like to thank the countless other Professors and fellow students that have helped me grow and learn throughout my time at UC Santa Cruz.

I. Introduction

Starting with the original routing protocol for the ARPANET [24], many routing protocols have been developed for networks operating as components of the Internet. These routing protocols were designed over twenty-five years ago, and were designed with the constraints of routing loops and hardware of the time in mind. This means that substantial improvements to many of these old protocols can be made. Storage and bandwidth have become abundant and cheap, and methods have since been conceived to solve the routing loop problem in a more efficient manner. This thesis focuses on transforming RIP (Routing Information Protocol) into a modern routing protocol taking advantage of the new technology available today and new methods to combat routing loops developed recently. This yields the new protocol proposed in this thesis, called Santa Cruz RIP (SCRIP). Comparisons are made between SCRIP and other interior gateway routing protocols like OSPF (Open Shortest Path First) and RIP, while not taking into account exterior gateway routing protocols like BGP (Border Gateway Protocol).

A difficult problem that must be considered when designing a routing protocol is the routing loop problem. Protocols like RIP suffer from long-term routing loops, other protocols like OSPF only experience temporary routing loops, and some like EIGRP (Enhanced Interior Gateway Routing Protocol) were designed to avoid routing loops. In each case, protocol design has been affected negatively because of the routing loop problem. RIP has long timers that limit the speed with which RIP can find paths to destinations, while OSPF has significant signaling and storage overhead costs because of the large amount of information that must be communicated reliably and tracked to avoid long-term routing loops. OSPF is a very complex protocol that would need major redesigning to fix the problems that it exhibits. On the other hand, RIP is a very simple protocol that does not have many complex operations, and hence additional features can be added to it without a negative impact to its core functionality. This is the motivation for this thesis to use RIP as a design baseline to create the protocol proposed in this

thesis. Santa Cruz RIP eliminates routing loop problems while also limiting the complexity and overhead that plagues past routing protocols.

The rest of this thesis is organized as follows. Chapter II describes some of the previous approaches to routing in wired and wireless networks. Key features of each protocol are described along with each protocol limitation. SCRIP builds on ideas from each of these protocols to create a routing protocol that is more efficient than any of the previous ones.

Chapter III introduces SCRIP, which provides loop-free least-cost paths to each destination in a computer network. Routes to destinations are learned using request, reply, and update messages both proactively and on-demand. Loop freedom is guaranteed through the use of reference distances included in each signaling message. A reference distance is used to limit which nodes are allowed to respond to requests and which routes are allowed to be added to the routing table of a node. A reference distance would be set equal to zero in a request when only the destination is allowed to respond. Multi-path routing is implemented by using distance tables so that multiple valid paths to a destination can be calculated. A hello protocol is used to establish connections with neighbors and detect link or node failures.

Chapter IV derives a formal proof of correctness to show that SCRIP is loop free at every point of its operation, and that SCRIP obtains least-cost path routes and eventually stops communicating changes to distances after topology changes in a network stop. The proof of loop freedom follows a similar proof to the one in [8], which also used reference distances to maintain loop freedom.

Chapter V describes the tests and performance metrics used to compare simulation results between RIP, OSPF, and SCRIP. It also discusses implementation details for each protocol addressed in these tests. Simulations are conducted using ns-3 [14]. The performance metrics that are measured in each test are time to convergence and amount of signaling messages sent. A variety of network topologies based on real world implementations are used in each test during different

scenarios that might occur in a network. These scenarios include important events including cold start, link and node failure, and link and node recovery.

Chapter VI discusses the results of the simulation experiments. The results indicate that SCRIP performs more efficiently in a majority of scenarios and would be preferable to both RIP and OSPF.

Chapter VII discusses SCRIP limitations and extensions to its current design and implementation for improved efficiency. It also describes future work that could be done to further improve the efficacy of the ideas presented in this thesis. Finally, Chapter VIII presents my conclusions.

II. Background and Motivation

The wired routing protocols used in the Internet today were created in the 1980s and 1990s, and not many updates have been made to these protocols since then. The routing protocols that this thesis focuses on are interior gateway routing protocols (IGPs) like RIP, OSPF, and EIGRP. Even though these protocols have evolved over the years, their inherent limitations have not been addressed. Chapter II-A shows an example of this evolution in RIP. This is a problem because each of these routing protocols exhibit various weaknesses that each stem from the way they have handled the routing loop problem. For example, RIP is slow to react to topology changes and OSPF uses a significant amount of bandwidth due to link state flooding. If the routing loop problem can be solved in a more efficient manner, then these problems with past protocols can be overcome without degrading the performance of the protocol in other areas. ODVR is a protocol that is able to solve the looping problem in wireless networks, and its principles can help shape a new wired routing protocol that can outperform these past protocols.

A. Routing Information Protocol

The Routing Information Protocol (RIP) seems like the prime example of a routing protocol to redesign and reconfigure. Many of RIP's design choices were

due to its routing loop problem, limited storage capabilities in routers, and low link bandwidth. These are all problems of the past and should not dictate the major mechanisms of a new routing protocol. The main drawbacks of RIP include routing loops, slow response to network changes, and incompatibility with large networks due to RIP's maximum hop count. This thesis uses RIP as a base to create a new routing protocol that fixes each of these problems. It is important to start by understanding RIP's history and why these problems exist in the first place.

RIP was first proposed and made into an internet standard in 1988 through Request For Comments (RFC) 1058 [13]. This initial version of RIP has many of the same features of the more recent versions of RIP including use of the Bellman-Ford algorithm, a maximum hop count, signaling message format, and optional split horizoning. RIPv2 was finalized in RFC 2453 [17], which added subnetting, basic security features, and some other basic enhancements. RIPv2 is the version of RIP that is implemented most often in networks today, and is the basis of conceptual changes and practical implementation in this thesis. However, RIPv2 still suffers from the same problems as the original RIP, and still operates the basic underlying shortest path algorithm. An RFC that adds on to RIPv2 is RFC 4822 [1], but again, these are just security enhancements and they do not solve the major problems seen in RIP. The newest version of RIP is RIPng, which supports IPv6 and was proposed in RFC 2080 [16].

For the rest of this thesis, whenever I refer to RIP, I am more specifically referring to RIPv2. The reason RIP suffers from routing loops is due to the lack of information that a RIP node tracks, and unfortunate timings of receptions of update messages immediately after link failure. This issue was noted in RFC 1058 and many attempts have been created in the past to solve these problems such as in [10], [11], [25]. Appendix A shows an example of how a routing loop can form in RIP. RIP suffers from a count to infinity problem that is a consequence of routing loops, which led to RIP's maximum hop count to attempt to cope with this problem. A secondary reason behind this limited hop count was to limit the

maximum size of routing tables to decrease storage and signaling costs. The slow time to convergence and reaction to topology changes are due to the timers in RIP. RIP waits a random period of time between one and five seconds before sending route update messages to neighbors, and waits up to 180 seconds before reacting to a node or link failure. This time period where a node has to wait to react to a topology change is called a dead timer. A periodic update is sent every thirty seconds, which allows a node to refresh these dead timers and include the most recent version of the routing tables. These choices for timer values were created to limit the chance for routing loops and limit link bandwidth. These timers are not suited for modern technology which is more reliable and can support high link bandwidth.

B. Open Shortest Path First Routing

A more reliable IGP is the Open Shortest Path First (OSPF) protocol. OSPF has its advantages and disadvantages, but simulations in multiple studies show that it outperforms RIP in almost all cases [2], [5], [15]. OSPF is a link-state routing protocol that is based on Dijkstra's shortest path algorithm. OSPF attempts to fix the routing loop problem by broadcasting topology information throughout the network, so that each node has a complete map of the network. This means that each node will know when a routing loop might form and can choose to route packets in such a way that routing loops are not possible. OSPF also has quicker times to convergence than RIP because it does not have to wait to send link state updates. OSPF's dead timer is set to forty seconds by default in most implementations [22], meaning that OSPF can detect and react to link and node failures much quicker than RIP. OSPF also makes use of a hello protocol that limits the amount of bandwidth needed to refresh the dead timers on each interface of a node, as well as allow for cooperation between nodes to send link state updates more efficiently.

These characteristics of OSPF make it seem that OSPF is always the best option, but OSPF does come with its disadvantages. The excessive broadcasting and complete topology stored at each node leads to significantly increased signaling and storage costs when compared to RIP. The hello protocol limits the signaling costs slightly, but the update and database description messages are very large because of the amount of information required to perform Dijkstra's algorithm properly. Link state updates are still sent out periodically to ensure that every node has the same view of the network, which again adds additional signaling overhead similar to RIP. Another downside is that transient loops are still possible in link-state routing protocols [18], and OSPF is no exception [7]. This means that during route computations, routing loops are still possible, however briefly. This thesis shows that a similar performance to OSPF can be achieved, at a much reduced cost in signaling overhead.

C. Enhanced Interior Gateway Routing Protocol

CISCO's Enhanced Interior Gateway Routing Protocol (EIGRP) is another wired routing protocol that attempts to eliminate routing loops and improve on the deficiencies of distance vector routing protocols like RIP. EIGRP was a CISCO proprietary product for much of its history, but was released and defined in RFC 7868 [23]. EIGRP makes use of the DUAL [9] algorithm to find loop free shortest cost paths in a network. EIGRP is a distance vector routing protocol like RIP, but it acts more like a hybrid routing algorithm due to the information stored and communicated by each node. EIGRP works by maintaining additional topology information at each node including successor information. This allows EIGRP to avoid routing loops and recover from link or node failure events safely. However, EIGRP performs poorly in large, hierarchical networks compared to OSPF. EIGRP also has higher storage and signaling overheads when compared to RIP.

D. Ordered Distance Vector Routing

Ordered Distance Vector Routing (ODVR) [8] was created to perform dynamic routing decisions in such a way that they are loop free. ODVR is a great example showing the enhancements that older IGPs should make use of to increase their own efficiency and performance, while no longer taking into account the restrictions of routing loops. ODVR was specifically created for use in mobile ad-hoc networks (MANETs). However, this does not mean that many of the same principles that ODVR uses cannot be employed in wired networks. ODVR uses reference distances to maintain an ordering of nodes in a network so that loops cannot form. This same idea is employed by SCRIP. Other features of ODVR that are ideal to adapt to wired networks include the idea of performing both on-demand and proactive routing, pending request tables, and multi-path routing mechanisms.

[8] proves that reference distances allow for loop freedom and shows, via simulation, the performance benefits that many of ODVR's additional features support over other MANET routing protocols like OLSR [3] and AODV [21]. If a port of ODVR to RIP would be possible, then RIP would no longer need to operate in such an inefficient manner because RIP would not have to deal with the routing loop problem. SCRIP shows where RIP can be made as efficient as possible because SCRIP does not have to contend with routing loop problems due to its use of reference distances. Chapter III details how some of the features of ODVR are implemented in SCRIP.

III. Santa Cruz RIP

This Chapter describes the operation of the proposed protocol, the Santa Cruz Routing Information Protocol (SCRIP). SCRIP is a loop free routing protocol that does not suffer from the deficiencies of previous wired routing protocols, like RIP and OSPF. SCRIP does this by using RIP as a base to build off of, and then modifies many of RIP's fundamental operations to create a distance vector routing

protocol that is loop free. The one thing that does not change is the underlying distance vector algorithm that allows for shortest path calculations. Appendix A gives an example of SCRIP's operation.

A. Loop Free Routing

SCRIP maintains loop free paths by maintaining reference distances to each destination in the network. Reference distances follow the logic introduced in [8]. This creates a strict ordering of nodes. Every request, reply, or update message contains a reference distance for each destination in the message. Upon reception of a request, nodes either answer the request if they have a reference distance to the destination that is less than or equal to the one in the request, or they will forward the request. A node trusts replies or updates only if the messages contain a reference distance that is less than or equal to the reference distance currently stored in the node. A request with a reference distance of zero means that the directly connected nodes of the destination are the only ones that can answer the request. This helps avoid routing loops when errors occur in the network, such as link and node failures.

B. Signaling Messages

The signaling messages in SCRIP are similar to those in RIP, however they are slightly modified to support reference distances and on-demand routing. The format for SCRIP signaling messages can be seen in Figure 1. Each message has a four byte header, along with a list of routing table entries (RTE). This format is exactly the same as in RIP, the difference is in how the RTEs are formatted to include reference distances. Figure 2 shows the format for an RTE in SCRIP. Each value in parentheses signifies the number of bytes the field takes up.

SCRIP uses the same message types as in RIP, but in a different way. SCRIP wants to maintain the small message sizes and relative format of the RIP RTE. There are multiple ways to do this, but the easiest way without losing any functionality is shown in Figure 2. In RIP, the distance (metric) field is simply four

Command (1)	Version (1)	Route Tag (2)
SCRIP Routing Table Entries (20)		
…		
SCRIP Routing Table Entries (20)		

Fig. 1: Santa Cruz RIP Signaling Message Format

Address Family Identifier (2)	Route Tag (2)
IP Address (4)	
Subnet Mask (4)	
Next Hop (4)	
Metric (2)	Reference Distance (2)

Fig. 2: Santa Cruz RIP Routing Table Entry

bytes. SCRIP splits this field in two where two bytes go towards the distance and two bytes go towards the reference distance. This is more than enough storage for each field because it means that SCRIP is able to track each destination up to 65535 hops away. Other ways to support reference distances are possible such as replacing the next hop field with reference distances. However, this would result in a loss of some functionality in the case in which three or more nodes are connected to a link at a time. A node would not be able to specify which neighbor should forward application traffic in these cases. The implementation for the simulations described in Chapter V follows what can be seen in Figure 2.

SCRIP uses four types of messages. The first is a request (REQ), which can be used in three ways. First, it can be used upon node startup to request the routing tables contained by each neighbor. These requests are broadcast messages that only ask the neighbors for their directly connected links. This is how they currently work in RIP and are only used to get the process of routing started. In RIP, they are used to tell each of their neighbors that they exist and result in an update message being sent by each neighbor to the requesting node. Second, if there is application traffic destined for a route that is not yet known by that node, then a REQ can be sent to attempt to learn that route faster than simply waiting for update messages to pass on the information. In this case, a reference distance of

infinity is specified so that any node that knows how to reach this destination can respond. As replies from various neighbors are received, the least-cost path route is installed in the requesting node. Third, they can be used to request routes to individual destinations. This represents the addition of on-demand routing to RIP which allows for faster recovery from failures in the network, along with allowing for loop free routes to be learned. This is the most common way in which a REQ is used, and occur during link and node failure and recovery events. During link failure, routes are invalidated that go through the interface that was connected to that link. A REQ is sent out each other interface to find the routes that were lost. These messages contain the destinations that previously went through this interface along with a reference distance (RD) of zero. This forces the destination to respond to the request. The reply of the destination along with the pending request tables (PRTs) ensure that the new route learned is a loop free shortest route. During link up events, requests are made for all known routes through this new link. It could be the case that a new least-cost path is through this new link and this route needs to be installed in the routing table. Again, a RD of zero is specified in these $REQs$. This functionality maintains loop free paths when state for certain routes to destinations are lost.

The second type of message is an update message (UPD). These messages allow for proactive routing in SCRIP, the same way that it was implemented in RIP. In RIP, these messages are called responses. Whenever a routing table is updated, a node sends a message to its neighbors notifying them of the updated routing table so that the neighbors can update their routing table. This process works very similarly to RIP, the only difference now is that the RD to each destination is also sent in each RTE. $UPDs$ that have RDs to destinations larger than what is stored in the node are not trusted to prevent a possible routing loop from forming. The third message type is a reply message (REP). A REP is a specialized UPD that is generated when a node receives a REQ and that node has a RD for the destination that is smaller than or equal to the the RD contained in the REQ.

$REPs$ are generally smaller than $UPDs$ because they only contain RTEs that were requested. A REP can be sent in immediate response, thereby satisfying this request. As the reply is received at each node, an entry for the request that this reply satisfies is looked up in the PRT. Using the PRT, the reply is forwarded out the interface that the request was originally received on. This forwarding process continues until the reply reaches the node that originally requested the reply. Along the forwarding path, a node can add the information stored in the reply if it represents a new shortest route to the destination that might have occurred due to link changes in the network.

The final message type was created to support a hello protocol. Large and infrequent periodic updates in RIP were inefficient and led to slow reaction to link and node failures. SCRIP can limit bandwidth and more quickly react to failures in the network by employing a hello protocol similar to the one used in OSPF and OLSR [3]. In SCRIP, that means that a special type of UPD can be used that can refresh the timers for all routes through a specific interface at the same time. This update message contains a single RTE with a well-defined format specifying a reserved IP address, so that a node can identify that this update is a hello message. Luckily, a multicast RIP address is already reserved and specified in RFC 2453 [17] that would work well for this purpose. Using the format of SCRIP messages, it can be seen that a hello message is only twenty-four bytes total, half the size of the hello messages used in OSPF. When a node receives a hello message, the node can refresh its dead timer for every route that goes through the interface that the hello message was received on. Whenever a change in a neighbor's routing table would occur would mean that an UPD would be sent. This means that these hello messages would only ever be refreshing the timers for valid routes, because these $UPDs$ have to be sent out with the most updated path information.

C. SCRIP Tables

SCRIP maintains three types of tables to perform its operations. These are the routing table, pending request table, and distance table. Entries within each table have a timeout period in which they are invalidated if not refreshed in a certain amount of time.

The first table is a routing table, which is a common data structure maintained by every routing protocol. A routing table contains a list of RTEs that represent the least-cost path route to each destination in the network. In SCRIP, it only stores information about neighbor nodes that are the next hop on the path to each destination. Routing tables are used to determine how application data and requests are forwarded throughout the network. The RTEs that are sent in $UPDs$ are the entries in the routing table.

The second table is the pending request table (PRT). A PRT is also a primary feature of ODVR [8], and allows for loop free routing and efficient request and reply signaling. A PRT is used to track which destinations have been recently requested, and where to forward replies to requests. In SCRIP, if one node makes a request for a particular destination and the neighbors cannot satisfy the request, this request gets added to the neighbor's PRT and then gets forwarded. There are two ways in which a request can be forwarded. If no information is known about a destination at a particular node, then the request gets flooded out of every interface except for the one the request was received on. This allows nodes in the network to find the destination the fastest when the nodes in the network do not yet have a route to the destination. A second forwarding method is when the node that receives a request has a route to the destination, but cannot satisfy the reference distance in the request. In this case, the request is only forwarded along this route so that it can get to a node that can satisfy the request's reference distance as efficiently as possible. We know that this is the best way to reach the destination and satisfy the request, since SCRIP maintains the least-cost path routes. This process of forwarding requests continues until a node can satisfy the

Destination	Mask	Recently Requested	Reference Distance	Interface List
10.0.0.0	255.255.255.0	False	0	1,2,3
10.0.5.0	255.255.255.0	True	0	1
10.0.5.0	255.255.255.0	False	1	2
10.0.9.0	255.255.255.0	False	5	1

Fig. 3: Santa Cruz RIP Pending Request Table Example

request. When the node can satisfy the request, a reply is sent out of the interface that the request was received on. From here, this reply is then forwarded back throughout the network to the original requestor by using the PRTs along the path. Replies are only be forwarded through the interfaces that the PRTs record to be the requestor.

The PRT also maintains a short timer so that it knows which routes were recently requested. If the timer has not yet expired, then that means we have requested a route to this destination recently and do not need to again because the reply has likely not had enough time to make its way back to the request forwarder yet. This can save bandwidth and reduce flooding of forwarded requests when a majority of nodes in the network do not know how to reach the destination. When requests for the same destination are received on a different interface, a new PRT entry is added and the request gets forwarded. If a reply for the destination is later received and satisfies the reference distances of both PRT entries, then the node can forward the reply out of both interfaces. As PRT entries are satisfied, they are deleted from the PRT. Figure 3 shows an example PRT that has received requests for multiple destinations.

The final type of table that is maintained is the distance table (DT), which allows for multi-path routing. Many routing protocols enact a multi-path routing mode that can help with network problems including congestion and link failures. This feature was not used in the original RIP protocol, and is an optional feature that can be turned on in SCRIP. This feature is implemented via a DT. A DT stores every route that it learns from its neighbors, even the ones that are not necessarily

Destination	Mask	Valid	Next Hop	Distance	Reference Distance	Interface
10.0.0.0	255.255.255.0	True	10.0.1.2	3	2	1
10.0.0.0	255.255.255.0	False	10.0.2.3	5	4	2
10.0.5.0	255.255.255.0	False	10.0.1.2	3	2	1
10.0.5.0	255.255.255.0	True	10.0.2.3	2	1	2
10.0.9.0	255.255.255.0	True	10.0.1.2	4	3	1
10.0.9.0	255.255.255.0	True	10.0.2.3	4	3	2
10.0.1.0	255.255.255.0	True	0.0.0.0	1	0	1
10.0.2.0	255.255.255.0	True	0.0.0.0	1	0	2

Fig. 4: Santa Cruz RIP Distance Table Example

the least-cost path. This means that every node will have multiple routes to each destination in the network. The number of routes for each destination is equal to the number of neighbors that each node has. What this essentially means is that each node will know how far away the destination is through each of its interfaces. Distance tables maintain the same RTEs that routing tables do, they just allow for every route to be learned instead of only the best route. Figure 4 shows an example distance table of a node that has two neighbors. It can be seen that each route, except the directly connected ones, have multiple paths to each destination. The routes that have the least-cost paths are deemed valid while the other ones are simply there for informational purposes.

A DT can be used to forward application data and help resolve link and node failures in the network. A validation scheme is used so that certain routes can either be valid or invalid. A route can only be valid if it is the best route to the destination, or it matches the best route's reference distance. This is to ensure that routing loops cannot occur with these secondary, valid entries. Valid entries can be used for making forwarding decisions and for replacing the best entry in the routing table upon link failure. When application data comes to a node, the node can use the DT to make a decision on how to forward the data. This feature can make decisions on how to forward the data in different ways based on multiple factors such as network congestion and using a scheme like ECN [6], at random, or something different. In this implementation and for the tests discussed in Chapter V, the scheme that is used is one where valid routes to the

destination in the DT are chosen at random, with a higher probability given to the valid route with the lowest reported distance. The second main reason DTs are used is for faster failure recovery. When a link fails, the distance table can be used to immediately swap in the next best valid entry. This means that convergence can be reached at the local node near instantly once again, and application data can continue to be forwarded through this new route.

D. Timers

The timer values in RIP were the biggest reason the protocol was slow to react to network changes and find convergence. The timers used in SCRIP are much smaller and appropriate considering the routing loop problem is no longer a factor in SCRIP. As mentioned previously, RIP's timers were long to limit the chance of a routing loop, but would not need to be if the routing loop problem was not a factor as in SCRIP and OSPF. This also means the maximum hop count in RIP is no longer necessary in SCRIP. Each message type has a different wait time specification before being sent. On-demand requests for destinations do not have any delay because the routes are needed immediately in these cases. Route replies also have zero delay to ensure that loop free routes are learned as soon as possible. Startup request messages still have the same one second delay as in RIP, because this is the time it takes for the node to start up and application traffic is not likely entering the node immediately upon startup.

SCRIP uses a configurable value for the delay of updates. For networks that have limited bandwidth, it might be a good idea to have some delay, while in other networks, this delay should be zero. The reason that the delay was so long in RIP was to allow multiple routing updates to be received, and to send as a batch update containing each new piece of information learned to be communicated at once. These delays would also reduce the chances of routing loops. In SCRIP, we no longer have to deal with these bandwidth or loop problems, so the default forwarding delay is zero. Experimentally, it was found that some delay might result

in more positive results than zero delay not only in terms of signaling messages sent, but also in times to convergence in some scenarios. This was because less messages had to be processed by each node meaning that many useless searches through the routing table were avoided when it was ultimately found that the same information was already known in the node.

The next important timer is the dead timer and the timer used for the hello protocol. RIP's dead timer is significantly longer than other routing protocols, causing longer wait times to adjust to network changes. Research has been done to find the ideal length of a dead timer in OSPF [12], but ultimately it seems that this value is too dependent on particular network conditions to have an optimal dead timer. The dead timer in SCRIP is set to forty seconds, matching the default used in OSPF implementations such as Quagga [22]. The timer for how often hello messages get sent is also matched to the default in OSPF. This timer specifies that a hello message is sent every ten seconds to each neighbor. Missing four hello messages in a row results in the protocol detecting a link or node failure. This allows for much faster detection of topology changes than the 180 seconds used in RIP.

IV. Proof of Loop Freedom and Correctness

A. *Proof of Loop Freedom*

The following theorems prove that SCRIP maintains loop free routing tables at every instant. Table 1 describes the notation used in each proof.

Table 1: Notation used in Loop Freedom Proofs

$\mathcal{T}$	Ordering Constraint
D_d^k	Distance from node k to destination d
s_d^k	Next hop of node k on the path to d
$D_{ds_d^k}^k$	Distance reported by next hop s of node k on the path to destination d
v_i	Arbitrary node i in the network

Theorem 1: A distributed routing algorithm (DRA) is loop free at every instant if the following ordering constraint is satisfied at every instant of every node's operation.

$$\mathcal{T} : D_d^k > D_d^{s_d^k} \tag{1}$$

Proof: Consider a set of nodes $\{v_1, v_2, ..., v_h, v_1\}$ different than destination d. Assume that this set of nodes create a routing table loop L of h hops by setting $s_d^{v_i} = v_{i+1}$ for $1 \leq i \leq h-1$ and $s_d^{v_h} = v_1$. It can be seen that this scenario leads to $D_d^{v_i} > D_d^{v_{i+1}}$ and $D_d^{v_h} > D_d^{v_1}$ for $1 \leq i \leq h-1$. This is a contradiction because it implies $D_d^{v_i} > D_d^{v_i}$.

This can also be seen another way. Assume, for the sake of contradiction, that a DRA is loop free, but at least one node does not satisfy $\mathcal{T}$. Assume this node that does not satisfy $\mathcal{T}$ is k and it has a neighbor j. This means that k would be allowed to set its next hop to a destination d equal to j, even if j sets its next hop to d as k. This is a contradiction, because this is a routing table loop between j and k. ■

Theorem 1 states that a routing protocol is loop free if it can satisfy $\mathcal{T}$ at every instant in the protocol's operation. In SCRIP, a route to a destination is assumed to have a finite distance. This means that our ordering constraint $\mathcal{T}$, can be rewritten as $\mathcal{O}$ in equation 2. If the route has a distance of infinity to the destination, then the path is assumed unreachable and a next hop to the destination is not designated. This means that routing will not take place. [8] proves that the ordering constraint in equation 2 satisfies the conditions for loop freedom described in Theorem 1.

$$\mathcal{O} : (D_d^i = \infty) \vee (D_{ds_d^i}^i < D_d^i < \infty) \tag{2}$$

If it can be shown that this ordering constraint is maintained at every point in a DRA's operation, then it can be said that the DRA is loop free. SCRIP maintain this ordering constraint via signaling messages and its operation described in Chapter

III. The proof of the following theorem demonstrates that SCRIP enforces $\mathcal{O}$ at every instant. This proof follows closely to what was shown in [8], but is slightly different due to some of the small differences between SCRIP and ODVR, and is included here for completeness.

Theorem 2: SCRIP ensures that $\mathcal{O}$ is satisfied by every node v_i for any given destination d at every instant.

Proof: $\mathcal{O}$ can be expressed in the following form:

$$\mathcal{O} = [\ D_d^i = \infty\ ] \vee [\ (D_{ds_d^i}^i < D_d^i) \wedge (D_d^i < \infty)\ ] \tag{3}$$

There can not be a value larger than infinity, therefore $\mathcal{O}$ can be simplified to the form seen in the first line of equation 4. Additional simplification can lead to the last line describing $\mathcal{O}$ in equation 4.

$$\begin{aligned} \mathcal{O} &\equiv [\ D_d^i = \infty\ ] \vee [\ (D_{ds_d^i}^i < D_d^i) \wedge \neg(D_d^i = \infty)\ ] \\ &\equiv ([D_d^i = \infty\ ] \vee [\ D_{ds_d^i}^i < D_d^i]) \wedge ([D_d^i = \infty] \vee [\neg(D_d^i = \infty)]) \\ &\equiv [\ D_d^i = \infty\ ] \vee [\ D_{ds_d^i}^i < D_d^i\ ] \end{aligned} \tag{4}$$

For the sake of contradiction, assume that every node executes SCRIP correctly and $\mathcal{O}$ is not satisfied at some moment t. This means that there must be at least one node v_i that executes the protocol correctly, most importantly the signaling overhead and its processing is executed properly. However, at time t, $\neg\mathcal{O}$ is true. From equation 4 and DeMorgan's law our assumption means that node v_i updates its routing state for destination d at time t with the condition given in equation 5.

$$\neg\mathcal{O} \equiv [\ D_d^i(t) < \infty\ ] \wedge [\ D_{ds_d^i}^i(t) \geq D_d^i(t)\ ] \tag{5}$$

From the operation of SCRIP, if node v_i does not have a route to the destination at time t, then $D_d^i(t) = \infty$. Therefore, it must be true that node v_i has routing state for d at time t so that $D_d^i(t) < \infty$ and $\neg\mathcal{O}$ in equation 5 can be satisfied. This means that node v_i can only select a node as its next hop to a destination such that $D_{s_d^i}^i(t) < D_d^i(t)$. This is a contradiction because this shows that SCRIP's

correct operation is counter to what is shown in equation 5. This means that this proof contradicts that SCRIP does not exhibit $\mathcal{O}$ at every instant in its operation. Therefore, the theorem is proved true. ■

Theorem 3: SCRIP maintains loop free routes at every instant.

Proof: The proof follows from Theorems 1 and 2. ■

B. *Proof of Correct Termination*

The following proofs show that SCRIP establishes shortest-path routes to each destination in a computer network within a finite time and stops communicating distance changes within a finite time after topology changes stop occurring. For the following proofs, a routing message is a signaling message that is used to find shortest-path routes. In SCRIP, these are request, reply, and update messages. Periodic messages used simply for neighbor detection are not taken into account. These proofs assume that each node executes SCRIP correctly, the maximum hop count between any two nodes is finite, and deadlocks cannot occur by the exchange of signaling messages.

Theorem 4: SCRIP attains routing tables that exhibit correct shortest-path routes to each destination in the network within a finite time after topology changes stop occurring.

Proof: The proof is by induction on the distance away from some destination i.

Basis Case: An arbitrary node becomes a neighbor of i and therefore has a distance to i of one. This node will receive hello and at least one update message from i. This is due to the operation of SCRIP in which hello messages are sent periodically between nodes and update messages are sent whenever a routing-table change occurs. Upon learning of a neighbor, the routing table would change and therefore an update to that neighbor would be sent. Given that each of these actions are controlled through finite timers in SCRIP, it follows that each node

would set its distance to its neighbor being equal to one within some finite time denoted by $T_0 \geq t$.

Inductive Step: Assume that SCRIP creates correct routes for all nodes $h-1$ hops away from the destination i. It must be shown that the proper route is obtained in a node that is h hops away from i. At some finite time $T_1 > T_0$, all nodes that are $h-1$ hops away from i have sent either an update or a reply to their neighbors stating that they are $h-1$ hops away from i. Each of these messages contain a reference distance that is either $h-2$ or lower to the destination i.

Suppose there is a node x that is h hops away from i. At some finite time $T_2 > T_1$, x must receive a reply or update from one of its neighbors stating that the neighbor is $h-1$ hops away from i. The reference distance contained in this message must be $h-1$ or lower. After reception of this update, the update is processed in x at a finite time $T_3 > T_2$.

At time T_3, only two cases may occur. In one case, node x may not need to change its distance because it already has a shortest-path route with a reference distance that is less than or equal to one plus what is stated in the routing message. In the second case, node x must accept the new route because the message reports a distance that results in a shorter route than its current distance and contains a reference distance that is smaller than what is currently maintained in the routing table of x. In the latter case, a route is created in x to i that has a distance of h and x selects a next hop that must have a distance to i of $h-1$ hops. This must be the case because SCRIP nodes always select routes through a next hop that contains a distance to the destination that is lower than its current reported distance and that satisfies its reference distance. By definition, a shortest path route must satisfy both of these conditions. Therefore, given the assumption that the route advertised to x is a shortest path, then the route that x has to i after processing the routing message must be a shortest path route. ■

Theorem 5: SCRIP stops sending messages reporting new distances within a finite time after topology changes stop occurring.

Proof: From the description of the operation of SCRIP in Chapter III, it follows that update, request, and reply messages can be generated only after processing an input event resulting from a topology change, or as a result of a message that forces the node to update its distance to a destination.

Given that all topology changes cease at some time t and the network is finite, there are only a finite number of topology changes and hence there can only be a finite number of update, request, and reply messages resulting from topology changes. Therefore, the only way in which a node may send a message is as a result of a message from a neighbor. However, it follows from Theorem 4 that all nodes in the network attain their shortest distances within a finite time T_0 after the occurrence of the last topology change taking place in the network. Accordingly, each node must send an update stating its shortest distance to each destination within a finite time $T_1 > T_0$. After that time, no node in the network can send a message reporting a new shorter distance to a destination or requesting a route to a destination, because there are no topology changes to process and no messages from neighbors that can change the shortest distance to any destination at any node. Therefore, the theorem is true. ■

Theorem 6: SCRIP computes shortest path routes and stops after some time t in which topology changes no longer occur in a network.

Proof: The proof follows from Theorems 4 and 5. ■

V. Methods for Evaluation

Chapters III and IV explain how SCRIP should be able to perform better than previous routing protocols. Simulation experiments were also conducted in order to show results that support these claims. The simulations give a comparison between RIP, OSPF, and SCRIP. Each simulation is run using ns-3 [14], specifically version 3.30. There is an existing implementation of RIP within ns-3 that was used to simulate RIP. ns-3 does not contain an implementation of OSPF, but does integrate Quagga's OSPF [22] into its framework, which was used for each simulation.

The default configurations for Quagga's OSPF were used. OSPF areas were also not used, which could change these results slightly. OSPF areas allows sections of the network to be aggregated together meaning that OSPF signaling could potentially be smaller since summary routes can be used to describe entire sections of the network. SCRIP was implemented by modifying the RIP implementation mentioned earlier.

A. Evaluation Metrics

The two main performance metrics that are measured in these simulations are the time to convergence and the signaling overhead used by each protocol. The time to convergence is defined as some quiet time in a network where the routing tables of each protocol can be observed to be unchanging for an amount of time. In these simulations, this quiet time is set to thirty seconds. This means that if the routing tables in each node are unchanged for thirty seconds, that we can declare convergence has been reached at the time where the routing tables first contained each of these routes. For example, if the routing tables had not seen an update for thirty seconds, and the simulation time is at forty seconds, then we know convergence was reached at ten seconds. Convergence was measured in this way because nodes in a network do not know when convergence is reached, but rather each node can only assume convergence has been reached when it stops seeing update messages. For example, RIP might create a routing loop and think it has reached convergence, but could know that it has not because routing updates would continue to be sent as the RIP nodes count to infinity. This method is acceptable for SCRIP and OSPF as well because routing tables would not change once each least-cost path route was found if there are no further topology changes. This means that new updates will also not be sent when there are no topology changes.

The signaling messages are counted in terms of number and size of packets sent by each protocol. An overhead signaling packet is defined as any message

exchanged between nodes that allows the protocol to operate and find least-cost paths. In RIP, these are request and response messages, but in OSPF these would be hello, link state packets, acknowledgements, etc. These tests want to understand the total bandwidth used by each routing protocol, so the total number of packets that get sent as signaling messages are counted. This means that if a packet is generated to be sent and is forwarded five times, then it counts as six signaling packets, because it consumed bandwidth six times.

B. Experimental Tests

Two different methods of testing were done. Each method uses a wide variety of network topologies based on real world implementations. The different test topologies are discussed in Chapter V-C. For each test, nodes are connected via a shared link with a two second delay and five megabit per second data rate value. The nodes share each link by running Carrier Sense Multiple Access (CSMA) across the shared medium. The first test measures how long it takes for convergence to be reached from cold start in each network topology, and measures the total signaling messages sent before and after convergence. This test runs for 500 seconds. This test was done to gain a baseline of results to see how each protocol behaves over a period of time when no topology changes occur. Application traffic is sent between hosts placed throughout each network. The goal of this test is to show the differences in signaling overhead when a network does not see topology changes.

The second type of test is based on [27]. The goal of this test is to understand each protocol's reaction to changing network conditions. These types of tests involve modifying the network in some way, such as cutting a link, and then measuring the amount of signaling messages it takes for a protocol to reach convergence based upon the new topology conditions. There are four types of topology changes that are measured within this test. They are link down, link up, node failure, and node recovery events. Link down and link up are simulated

by disabling or enabling both interfaces connected to a link. Node failure and recovery are simulated by either disabling or enabling all interfaces on the node that is being modified. Each event is measured separate from each other. For each test in this thesis, tests are run multiple times with an average of the results being taken. For example, response to link failure would be observed when different links fail. RIP's timers are based on some randomness as mentioned in Chapter II, so simulations were run multiple times with different random seeds to find a mean response of RIP to each test.

C. Test Topologies

A variety of real world test topologies are used to evaluate the performance of each routing protocol. As was found in [27], a routing protocol is not always the best choice in every single situation. Some routing protocols perform better in specific topologies versus others. Each of these test topologies were taken from actual network implementations, except one that was inspired by a real topology, but slightly modified to create major routing table changes when certain links and nodes fail. Resources used to find each of these topologies and a host of other real world implementations can be found at [4], [26], [27].

The test topologies are shown in Figures 5 - 9. These particular topologies were picked because there was potential for routing loops and multiple paths to each destination. Many of the implementations that were found through the sources mentioned earlier were star topologies or similar. These topologies have less opportunity for routing decisions to be made and routing loops can only occur between two nodes, which RIP already protects against via split horizoning or poison reverse. The topologies chosen have possibilities for loops of three or more nodes and force nodes to make the best routing decisions since more than one path to each destination exists. Each link cost is given equal weight to give the most equal comparison between OSPF, RIP, and SCRIP. This is because the distance metrics are different in OSPF versus RIP and SCRIP, which could produce

drastically different results based upon link cost values. OSPF uses bandwidth of the link as its distance metric, so the case could be that the longest path in terms of hop count would actually be the fastest path to forward traffic because the link bandwidth is so low on the path that is least-cost in terms of hop count.

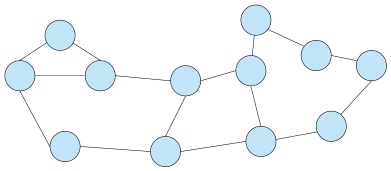

Fig. 5: Abilene Network Topology [26]

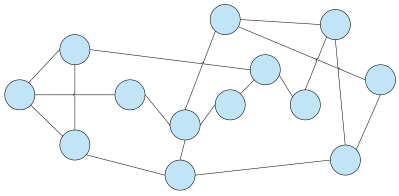

Fig. 6: NSFNET-T1 Network Topology [27]

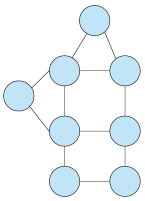

Fig. 7: UK-Backbone Network Topology [4]

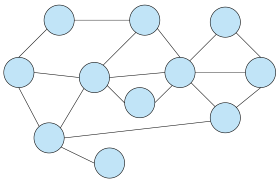

Fig. 8: Custom Test Network Topology

D. SCRIP Implementation

SCRIP is implemented in ns-3 by changing many of the fundamental operations of the RIP code for comparison in these simulations. The implementation of SCRIP is as described in Chapter III. New data structures had to be created

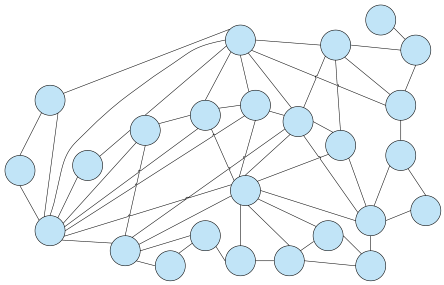

Fig. 9: ATT North America Network Topology [26]

to support the pending request tables and distance tables. The RIP routing table entries also needed to change to support reference distances, as seen in Figure 2. The timers had to be updated to match what was described in Chapter III-D. The formats for hello, request, and reply messages were added. Finally, the processing of each type of message and the addition of routes into the routing tables, distance tables, and pending request tables were added. The implementation and the simulations used can be found at [20].

VI. Results

This Chapter outlines the results of each of the tests and simulations that were described in Chapter V. The results for each of the tests can be seen in Tables 2 - 6. The convergence is the amount of time it takes from the event in the experiment until least-cost paths to every destination in the network are known at every node. This value is measured in seconds. Packet count is the total number of signaling messages that use bandwidth. Packet size is the total size of each of those signaling messages measured in bytes. The most important metric is the time to convergence, with a secondary emphasis being the size of all signaling messages, since this is the total bandwidth that is being used by the protocol.

Table 2 shows the results of the cold start experiment. It can be seen that OSPF and RIP both find convergence significantly slower than SCRIP. This is due to the fast message forwarding in SCRIP and also the requests made for destinations in the network. OSPF appears to perform worse than RIP in terms of time to

Table 2: Cold Start Experiment Results

Metric	RIP	OSPF	SCRIP
Abilene Cold Start Scenario			
Convergence	49.4	41.2	1.5
Packet Count	615.1	15,576.8	1,630.7
Packet Size	181,102.3	1,400,320.9	67,666.6
NSFNET-T1 Cold Start Scenario			
Convergence	46.9	49.4	1.6
Packet Count	708.6	20,179.5	1,898.8
Packet Size	246,736.1	2,173,246.3	84,760.1
UK-Backbone Cold Start Scenario			
Convergence	44.3	46.2	1.5
Packet Count	430.2	12,599.8	1,192.5
Packet Size	100,484.8	1,981,691.4	45,702.0
Custom Topology Cold Start Scenario			
Convergence	44.7	50.3	1.8
Packet Count	638.1	26,523.7	1,903.1
Packet Size	225,182.1	5,442,330.7	83,278.3
ATT North America Cold Start Scenario			
Convergence	50.1	53.8	2.4
Packet Count	2,098.6	214,764.5	5,455.2
Packet Size	2,025,252.3	48,258,920.1	455,896.4

Table 3: Link Down Experiment Results

Metric	RIP	OSPF	SCRIP
Abilene Link Down Scenario			
Convergence	45.2	1.4	1.7
Packet Count	98.6	200.1	623.6
Packet Size	12,112.0	19,004.8	22,390.6
NSFNET-T1 Link Down Scenario			
Convergence	44.3	2.9	1.8
Packet Count	236.4	246.2	513.4
Packet Size	29,799.8	22,248.1	22,020.9
UK-Backbone Link Down Scenario			
Convergence	45.7	1.6	1.4
Packet Count	119.1	171.9	224.6
Packet Size	22,576.5	12,872.4	7,810.0
Custom Topology Link Down Scenario			
Convergence	42.2	9.3	1.6
Packet Count	255.0	785.2	605.7
Packet Size	30,340.5	117,820.1	20,956.7
ATT North America Link Down Scenario			
Convergence	45.4	9.1	2.1
Packet Count	381.4	3,582.3	3,689.5
Packet Size	199,684.2	762,900.7	129,310.8

Table 4: Link Up Experiment Results

Metric	RIP	OSPF	SCRIP
Abilene Link Up Scenario			
Convergence	46.7	1.5	1.9
Packet Count	127.5	162.8	288.3
Packet Size	22,448.3	12,416.5	10,008.9
NSFNET-T1 Link Up Scenario			
Convergence	46.9	4.2	1.3
Packet Count	141.1	503.4	330.1
Packet Size	26,584.9	44,040.7	10,632.7
UK-Backbone Link Up Scenario			
Convergence	56.9	1.2	1.6
Packet Count	117.1	101.7	119.8
Packet Size	15,348.3	18,968.4	3,884.3
Custom Topology Link Up Scenario			
Convergence	50.1	9.3	1.9
Packet Count	132.1	1,263.3	215.6
Packet Size	30,972.4	171,040.8	7,292.6
ATT North America Link Up Scenario			
Convergence	63.1	9.2	2.8
Packet Count	433.7	3,583.1	1,277.7
Packet Size	307,912.1	715,284.5	45,654.5

Table 5: Node Failure Experiment Results

Metric	RIP	OSPF	SCRIP
Abilene Node Down Scenario			
Convergence	222.8	43.6	41.7
Packet Count	155.2	907.6	662.7
Packet Size	54,080.1	79,632.1	22,626.7
NSFNET-T1 Node Down Scenario			
Convergence	223.7	45.1	41.7
Packet Count	382.5	1,293.1	1,677.0
Packet Size	104,668.3	128,880.7	53,268.1
UK-Backbone Node Down Scenario			
Convergence	213.0	42.9	40.9
Packet Count	193.0	763.7	469.0
Packet Size	38,432.3	117,116.6	15,220.4
Custom Topology Node Down Scenario			
Convergence	211.7	47.5	41.1
Packet Count	298.3	1,569.4	1,049.3
Packet Size	83,592.2	315,308.1	34,693.9
ATT North America Node Down Scenario			
Convergence	217.1	50.3	42.9
Packet Count	1,313.1	23,787.4	10,809.7
Packet Size	909,812.5	3,003,204.9	386,068.6

Table 6: Node Recovery Experiment Results

Metric	RIP	OSPF	SCRIP
Abilene Node Up Scenario			
Convergence	31.3	5.3	1.9
Packet Count	92.3	369.0	353.1
Packet Size	19,756.4	23,280.8	14,260.5
NSFNET-T1 Node Up Scenario			
Convergence	31.9	6.1	1.7
Packet Count	109.9	693.1	598.3
Packet Size	25,554.0	53,956.5	26,332.4
UK-Backbone Node Up Scenario			
Convergence	27.1	1.3	1.6
Packet Count	69.7	382.5	201.1
Packet Size	12,016.8	37,524.1	8,476.4
Custom Topology Node Up Scenario			
Convergence	33.0	4.8	1.5
Packet Count	94.4	2,351.2	282.9
Packet Size	19,285.5	219,448.1	12,528.0
ATT North America Node Up Scenario			
Convergence	39.1	8.8	2.8
Packet Count	503.5	41,351.8	1,867.4
Packet Size	202,732.4	4,930,040.2	113,476.6

convergence, but this is because OSPF sends out four rounds of hello messages in order to understand neighbors and designate nodes to advertise link states. This means that convergence is not attempted to be reached until forty seconds into each simulation. When taking this factor into account, it makes the convergence results look much more comparable to SCRIP rather than RIP. Once convergence is reached, each protocol sends out periodic messages to detect topology changes and ensure the most recent route information is known. RIP would expect to have a small signaling overhead, but this is not the case because the periodic updates that are sent after convergence contain the entire routing table. OSPF performs the worst of the three in terms of signaling overhead because it periodically sends out link state updates and hello messages to make sure that the most recent information is known throughout the network. SCRIP only uses hello packets, which are only twenty-four bytes large each, to detect if any link changes have occurred. The benefits of this method are obvious because SCRIP's total signaling overhead is a fraction of the size of both RIP and OSPF. The number of packets in SCRIP is higher than RIP, but this is because RIP sends out large periodic updates every thirty seconds, versus the small hello packets which are sent every ten seconds in SCRIP.

Tables 3 and 5 show the results for the link down test and node down test, respectively. The results are easy to understand because SCRIP is able to recover much quicker and reach convergence faster by using its request messages. RIP propagates the link changes very slowly due to its timers leading to long recovery times. OSPF updates the topology in a similar way, but does not have timers that limit how fast it can send the updates throughout the network. SCRIP recovers much faster because it sends requests for specific destinations that were lost due to the link or node failure, and can receive replies very quickly because there are no limitations on how fast these messages can be sent. The signaling overhead is much lower in SCRIP because the protocol is able to recover very quickly, but it can be seen that the signaling overhead is actually higher in SCRIP when OSPF

reaches convergence faster. OSPF's recovery times are faster in the two smallest networks due to the limited number of nodes that the link state information must be shared with. SCRIP performs better in larger networks compared to OSPF. This is because OSPF uses flooding with each of its updates, which increases the total propagation delays of packets around the network along with the total processing times of all of the packets especially as networks get larger and more flooding is required. On the other hand, SCRIP requests and replies are mostly sent in unicast fashion throughout the network during recovery. The messages are also much smaller than OSPF's because each request and reply is only be twenty-four bytes large, compared to the much larger link state updates in OSPF. The large discrepancy with RIP's recovery time in the node down test is because RIP's dead timer is set to 180 seconds, while SCRIP and OSPF detect the node failure after forty seconds.

Tables 4 and 6 exhibit similar results. That is that OSPF performs better in the smaller topologies, while SCRIP is able to limit the signaling overhead and reach convergence faster in the larger topologies. These results follow what was found in [27], which found that certain routing protocols perform better than others depending on the situation and network topology. Again, RIP must wait a long time before it is allowed to share the information of the new link, and OSPF must flood this new link information throughout the network. This results in RIP exhibiting slow times to convergence and OSPF using a high amount of bandwidth, as expected. SCRIP is more efficient in both cases by making use of requests to find the least-cost path routes through links that have newly become available.

VII. Protocol Limitations and Future Work

This Chapter describes two areas of potential future work with SCRIP and the idea of using reference distances for loop free routing. They are in terms of protocol improvements and testing. SCRIP is a promising new wired routing

protocol for Internet networks, but improvements are possible if sacrifices are willing to be made in other areas. SCRIP uses hop count as its distance metric for both distance and reference distance to a destination. A scheme similar to the one used in EIGRP could be employed that would create a distance value based on link bandwidth, delay, load, etc. This could improve how routes are found, so that the better routes can be found based on particular network conditions, rather than just hop count. Another pro would be better handling of link cost changes. Since SCRIP uses hop count as its distance metric, it does not route based on the quality of a link, meaning that a path that is slower to the destination might be chosen because there are fewer hops. However, this would add significant complexity to the protocol and would require additional information to be added within the signaling messages of SCRIP. Currently a method has not been devised that could use these metrics with reference distances properly. Another potential area for improvement could be the implementation of a hierarchical scheme like the one that is in OSPF. Certain nodes could be configured to be border nodes that would send out smaller routing table updates that would summarize an entire area of a network. Again, this would add additional complexity and potentially change the format of certain signaling messages.

SCRIP has been shown to perform better than RIP and OSPF, but this is just a subset of the wired routing protocols used in the Internet today. In future simulations, comparisons should also be made versus BGP and a DUAL based protocol, like EIGRP. BGP would be an important protocol to test against to understand if SCRIP has potential to work in Internet backbone networks. EIGRP would also be a good data point to test against because it is another distance vector protocol that maintains loop freedom. Along with testing against other protocols, additional metrics would also be good to test. Some ideas include storage costs, processing time of signaling packets and algorithm computation, and effect of hop count as a distance metric.

VIII. Conclusion

This thesis introduced Santa Cruz RIP (SCRIP), a loop-free routing protocol that limits the complexity and overheads of past interior gateway routing protocols. SCRIP builds upon the distance-vector foundation of RIP, and incorporates ideas from other routing protocols like OSPF and ODVR. SCRIP introduces a routing method that is loop free by stating reference distances in route requests, replies, and updates. On-demand and proactive routing ideas are used in conjunction to ensure that least-cost path routes to destinations are obtained reliably and quickly. This thesis proved that SCRIP is loop free at every instant and that it terminates correctly. Simulations were run showing that SCRIP performs better than RIP and OSPF in terms of convergence times and the amount of signaling overhead used in each protocol.

APPENDIX A
EXAMPLE RIP AND SCRIP OPERATION DURING LINK FAILURE

This appendix shows an example of how RIP might create a three or more node loop during link failure. It then gives an example of SCRIP in the same scenario to display the basic operations within SCRIP, and to show how loop freedom is maintained.

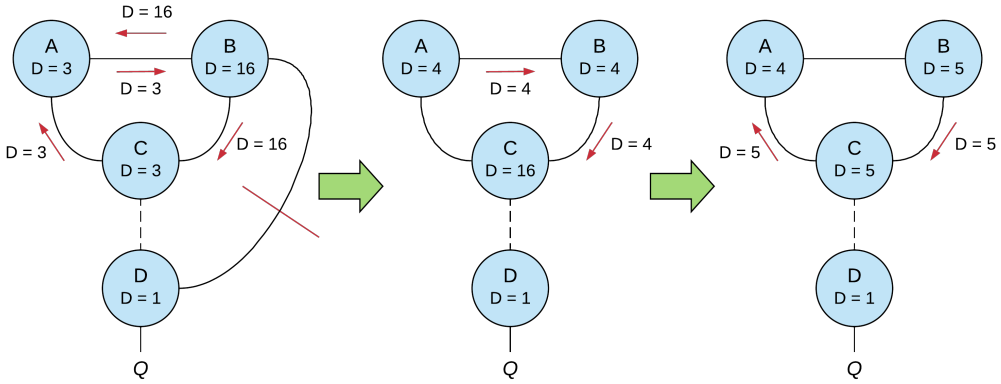

Fig. 10: How a three node loop can form in RIP during link failure

Refer to Figure 10 for the following explanation on how a loop can form between three or more nodes in RIP. Each node in this network is labeled $A, B, C,$ or D. All labels that are in the form D = x, where x is a number, represent the distance from the node to the destination Q. Each link has a weight of one, except the dashed lines between node C and D. This is a simulated connection that passes through at least three other nodes in the network. For this example, assume there are ten nodes between C and D, meaning that there is a distance of eleven hops to get from C to Q. The arrows between nodes represent update messages that get sent as a result of a change to the routing table entry (RTE) stored in the routing table corresponding to destination Q. Each green arrow represents a shift in time between the different events occurring.

The most left topology shows the scenario when a link fails between B and

D, meaning that B sets its distance to destination Q to infinity (16 in the case of RIP). At this time, node B sends an update to its neighbors that it can no longer provide a path to Q. A periodic update from A is received right after the change in distance to 16 in B while an update from C to A is received right after A learned that it can no longer route through B. In this case, B sets its next hop to Q as A, A sets its next hop to Q as C, and C sets its next hop to nothing since it cannot route through B any longer. The middle topology shows that A and B communicating their new knowledge, while C has to wait longer to tell A that it can no longer be its next hop to Q due to a longer random timer. This results in each node updating its distance to Q because its next hop is reporting a new distance. Finally, in the right topology, it can be seen that this process will continue and the nodes will continue to update their distances and count to infinity. In this case it would only be eleven because that is the point at which C would choose to use the path through the dashed line to D, but it would be easy to understand that if this path did not exist then they would all count to infinity.

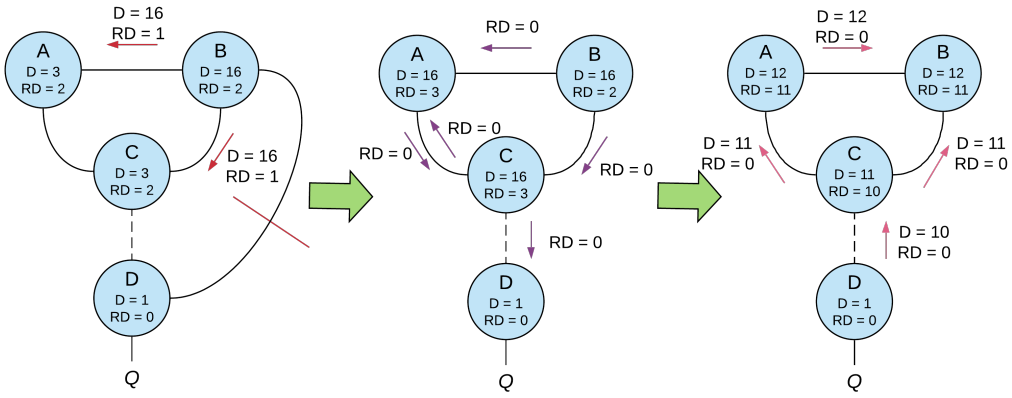

Fig. 11: Example of SCRIP's operation during link failure

Now the same scenario is discussed for SCRIP. Refer to Figure 11 for the following explanation. The setup is exactly the same, except now distance and reference distance (RD = y) is tracked, and request and reply messages are also in use. Some updates that might occur in the network are omitted, but that is

because they would not be trusted anyways since the RD would not satisfy the value stored in the node receiving the update. I explain these cases as they could come up. Although SCRIP does not have a maximum hop count, 16 is used as infinity to be consistent with the definition in the RIP example.

At the time shown in the leftmost topology, the link fails between C and D, changing the distance and reference from two to 16 and one to two respectively in node B. This results in an update message being sent by B to A and C with this information. Even if updates had been sent by A and C towards B at this time, B would not accept the route because the incoming reference distance of three would not be trusted.

Immediately after this update is sent, a request is sent by B to find a new route to Q. As A and C receive the updates, they would also send requests for Q out of each interface except the one towards B because B just updated them that it did not know how to reach Q. As each of these requests are received at the nodes A and C, they would forward them along. This can be seen in the middle topology of Figure 11. The requests from B were sent before the ones from A or C, and could have been forwarded before the ones generated by A and C. At this point, B's PRT contains one entry that it itself generated. A's PRT contains one entry, but it specifies three interfaces for this entry. Each interface corresponds to one of its neighbors or itself. Each request received at A was for Q with a RD of zero, so they can all be grouped together. C's PRT is very similar to A's but C was also able to forward the requests through the dotted path towards D.

Each node on the dotted path forwards the request towards D, which can answer the request because it has a RD of zero to Q. The right topology shows D sending a reply in response to the request and that reply being forwarded back through the network to each node that originally requested the route to Q. As $A, B,$ and C receive this reply, they consult their PRT's and forward the reply out of each interface corresponding to the entry that the reply satisfies. It should be noted that the distance increases as the replies are forwarded from hop to hop,

but the reference distance remains the same. A might think to forward the reply to C since it originally got a request from C, but it does not because its next hop to the destination is through C, and sending the reply could result in a routing loop. The final values for distance and reference distance contained within each node of the rightmost topology is the result of receiving each of the replies. The reference distance is one less than the distance in the final conclusion and not zero because it uses this value to update its neighbors in the update messages that would be exchanged after these new routes are installed. If the link would suddenly fail between C and D, and A had a RD of zero, then C would believe A even though this could lead to a routing loop. At this point, convergence has been reached and no new update messages would need to be sent after the update that was sent as a result of reception of the reply.

References

[1] Atkinson, R. and M. Fanto, "RIPv2 Cryptographic Authentication" *RFC 4822*, 2007.

[2] L. D. Circiumarescu, G. Predusca, N. Angelescu, and D. Puchianu, "Comparative Analysis of Protocol RIP, OSPF, RIGRP and IGRP for Service Video Conferencing, E-mail, FTP, HTTP," in 2015 20th International Conference on Control Systems and Computer Science, May 2015, pp. 584–589, doi: 10.1109/CSCS.2015.17.

[3] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)" *RFC 3626*, 2003.

[4] M. Dodge, "An Atlas of Cyberspaces- ISP Backbone Maps," Maps of Internet Service Provider (ISP) and Internet Backbone Networks, 2004. https://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/isp_maps.html.

[5] C. G. Dumitrache, G. Predusca, L. D. Circiumarescu, N. Angelescu, and D. C. Puchianu, "Comparative study of RIP, OSPF and EIGRP protocols using Cisco Packet Tracer," in 2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE), Oct. 2017, pp. 1–6, doi: 10.1109/ISEEE.2017.8170694.

[6] S. Floyd, "TCP and explicit congestion notification," SIGCOMM Comput. Commun. Rev., vol. 24, no. 5, pp. 8–23, Oct. 1994, doi: 10.1145/205511.205512.

[7] P. Francois and O. Bonaventure, "Avoiding Transient Loops During the Convergence of Link-State Routing Protocols," IEEE/ACM Transactions on Networking, vol. 15, no. 6, pp. 1280–1292, Dec. 2007, doi: 10.1109/TNET.2007.902686.

[8] J. J. Garcia-Luna-Aceves and E. Hemmati, "ODVR: A Unifying Approach to On-Demand and Proactive Loop-Free Routing in Ad-Hoc Networks," 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 2019, pp. 1-11.

[9] J. J. Garcia-Luna-Aceves, "Loop-free routing using diffusing computations," IEEE/ACM Trans. Networking, vol. 1, pp. 130–141, Feb. 1993.

[10] J. J. Garcia-Luna-Aceves and Shree Murthy, "A path-finding algorithm for loop-free routing," IEEE/ACM Transactions on Networking, vol. 5, no. 1, pp. 148–160, Feb. 1997, doi: 10.1109/90.554729.

[11] J. J. Garcia-Luna-Aceves and S. Roy, "On-demand loop-free routing with link vectors," IEEE Journal on Selected Areas in Communications, vol. 23, no. 3, pp. 533–546, Mar. 2005, doi: 10.1109/JSAC.2004.842564.

[12] M. Goyal, K. K. Ramakrishnan, and Wu-chi Feng, "Achieving faster failure detection in OSPF networks," in IEEE International Conference on Communications, 2003. ICC '03., May 2003, vol. 1, pp. 296–300 vol.1, doi: 10.1109/ICC.2003.1204188.

[13] C.L. Hedrick, "Routing Information Protocol" *RFC 1058*, 1988.

[14] T. R. Henderson, M. Lacage, and G. F. Riley, "Network Simulations with the ns-3 Simulator," p. 1.

[15] S. Y. Jalali, S. M. Wani, and M. A. Derwesh, "Qualitative Analysis and Performance Evaluation of RIP , IGRP , OSPF and EGRP Using OPNET TM," 2014.

[16] G. Malkin and R. Minnear, "RIPng for IPv6" *RFC 2080*, 1997.

[17] G. Malkin, "RIP Version 2" *RFC 2453*, 1998.

[18] J. M. McQuillan, I. Richer, and E. C. Rosen, "The New Routing Algorithm for the ARPANET," Ieee Transactions on Communications, vol. 28, no. 5, 1980.

[19] J. Moy, "OSPF Version 2" *RFC 2328*, 1998.

[20] S. Neuschmid, "Santa Cruz RIP Repository", 2020, Bitbucket Repo, https://bitbucket.org/sneuschmid/santa-cruz-rip-repository

[21] C. E. Perkins and E. M. Royer, "Ad-Hoc On-Demand Distance Vector Routing," *Proc. IEEE WMCSA '99*, 1999.

[22] "Quagga: OSPFv2." https://www.nongnu.org/quagga/docs/docs-multi/OSPFv2.html (accessed Apr. 30, 2020).

[23] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, R. White, "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)" *RFC 7868*, 2016.

[24] M. Schwartz and T. Stern, "Routing Techniques Used in Computer Communication Networks," IEEE Transactions on Communications, vol. 28, no. 4, pp. 539–552, Apr. 1980, doi: 10.1109/TCOM.1980.1094690.

[25] Ch. Steigner, H. Dickel, and T. Keupen, "RIP-MTI: A New Way to Cope with Routing Loops," in Seventh International Conference on Networking (icn 2008), Apr. 2008, pp. 626–632, doi: 10.1109/ICN.2008.58.

[26] The University of Adelaide, "The Internet Topology Zoo," The Internet Topology Zoo, 2013. http://www.topology-zoo.org/index.html.

[27] William Zaumen and J. J. Garcia-Luna Aceves. Dynamics of distributed shortest-path routing algorithms. In Proceedings of the conference on Communications architecture and protocols, 1991 August, pp. 31-42.