3-26-2015

Airborne Network Optimization with Dynamic Network Update

Bradly S. Paul

AIRBORNE NETWORK OPTIMIZATION WITH DYNAMIC NETWORK UPDATE

THESIS

Bradly S. Paul, Capt, USAF

AFIT-ENG-MS-15-M-030

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-15-M-030

AIRBORNE NETWORK OPTIMIZATION WITH DYNAMIC NETWORK UPDATE

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Bradly S. Paul, B.S.C.P.

Capt, USAF

March 2015

AFIT-ENG-MS-15-M-030

AIRBORNE NETWORK OPTIMIZATION WITH DYNAMIC NETWORK UPDATE

Bradly S. Paul, B.S.C.P.
Capt, USAF

AFIT-ENG-MS-15-M-030

Abstract

Modern networks employ congestion and routing management algorithms that can perform routing when network routes become congested. However, these algorithms may not be suitable for modern military Mobile Ad-hoc Networks (MANETs), more specifically, airborne networks, where topologies are highly dynamic and strict Quality of Service (QoS) requirements are required for mission success. These highly dynamic networks require higher level network controllers that can adapt quickly to network changes with limited interruptions and require small amounts of network bandwidth to perform routing. This thesis advocates the use of Kalman filters to predict network congestion in airborne networks. Intelligent agents can make use of Kalman filter predictions to make informed decisions to manage communication in airborne networks. The network controller designed and implement in this thesis will take in the current and predicted queue size values to make intelligent network optimization decisions. These decisions will enhance the overall network throughput by reducing the number of dropped packets when compared with current static network and MANET protocols.

Acknowledgements

I would like to thank my beautiful wife and daughter for their unrelenting support during this Masters Degree effort. I would also like to express my sincere appreciation to my faculty advisor, Maj Dube, for his guidance and support throughout this entire experience. I would also like to thank my committee members, Dr. Hopkinson and Dr. Mullins for their support throughout this research effort.

Bradly S. Paul

Table of Contents

Page

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
Mbps	Megabits per second	29
IP	Internet Protocol	33
DNUC	Dynamic Network Update Controller	35
kb	kilobits	35
RTT	Round Trip Times	35
MTU	Maximum Transmission Unit	43
MANETs	Mobile Ad-hoc Networks	1
QoS	Quality of Service	1

AIRBORNE NETWORK OPTIMIZATION WITH DYNAMIC NETWORK UPDATE

I. Introduction

Today, more than ever, the United States (US) military relies heavily on airborne networks to transmit critical battlespace information from air vehicles to commanders on the ground. As the number of airborne vehicles increases, as does the total amount of information collected. With this increasing amount of collected information, it becomes critical to pass this information to the commanders on the ground to ensure they have the Situational Awareness (SA) to make informed decisions. To keep up with this demand, network optimization becomes more important to ensure all information collected is transmitted to its destination. Current routing algorithms and queue prediction schemes are sufficient for today's Internet and corporate networks but they lack the flexibility and adaptive qualities airborne networks require. These networks require sufficient optimization in order to transmit data quickly from node to node to allow military leaders to make informed decisions in their respective missions. Today's networking schemes lack the flexibility and adaptive qualities necessary to make immediate routing decisions necessary for airborne networks. The lack of flexibility often results in inadequate performance, increased packet loss, and a loss of overall network throughput. Prediction tools and the intelligent agents collecting network information could be used to reduce packet loss and increase overall network throughput.

1.1 Background

Past research in the area of airborne networks and network flows has made some progress but many issues are still unresolved. With the dynamic qualities that airborne networks possess, the network components must be able to support ever changing needs. Research in this area has concentrated on network flows [8] and router queue size prediction [15] to predict when and where changes occur and how to route network traffic through a network more efficiently and with greater reliability [7]. These intelligent agents have utilized perception to discern current network conditions and act accordingly with limitation [29]. Past research was limited to simulations of small-scale, land-based networks [15] instead of emulating networks with variable link speeds as seen in airborne MANETs.

To ensure that the US is able to maintain a superior war-fighting capability, the Department of Defense (DoD) continuously explores the latest networking technologies. The DoD recently invested in the Global Information Grid (GIG) shown in Figure 1.1, which the National Security Agency (NSA) defines as "the globally interconnected, end-to-end set of information capabilities for collecting, processing, storing, disseminating, and managing information on demand to war-fighters, policy makers, and support personnel" [3]. "The GIG is a net-centric system operating in a global context to provide processing, storage, management, and transport of information to support all DoD . . . missions and functions." It also states "GIG capabilities are available from all operating locations: bases, posts, camps, stations, facilities, mobile platforms, and deployed cites [3]. With this in mind, the GIG boundary shown in Figure 1.1 refers to the separation of the deployed network and the GIG itself while ensuring that the information in the GIG is available to the deployed network.

Aircraft bandwidth is directly proportional to the distance between the two nodes. For instance, as distance between two aircraft increases, the bandwidth continues to

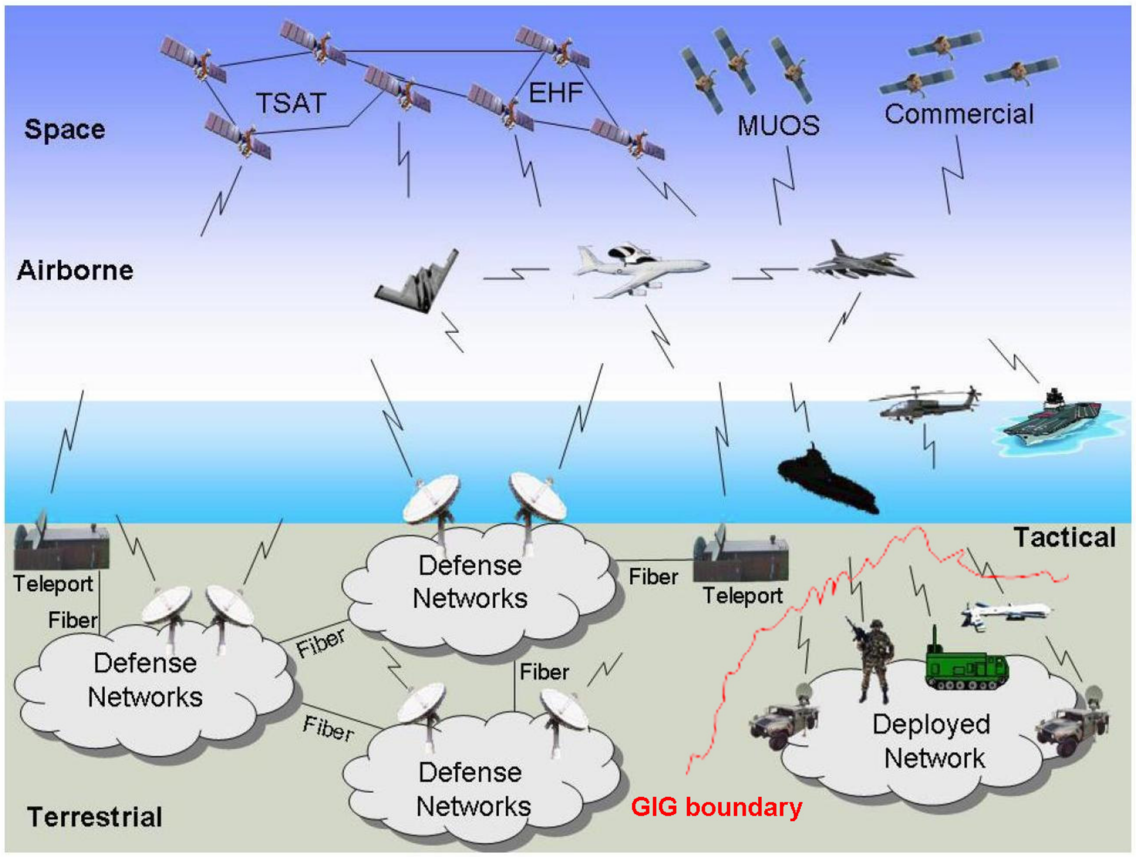


Figure 1.1. Global Information Grid [2]

decrease until the distance becomes too great for the connection to stay alive. On the other hand, as two aircraft approach one another, the connection speed between them increases. During military operations, as an aircraft's position changes throughout the battlespace, the bandwidth between other aircraft changes as well. This constantly changing bandwidth creates a highly dynamic airborne network. The dynamic nature of this type of network requires a queue size prediction scheme to determine when links could drop and forward this information to control algorithms, thus allowing these algorithms to make more informed decisions on how to route traffic throughout the network.

Typically, commanders plan tactical airborne environments at least a day in advance, in a document called an Air Tasking Order (ATO). This information can be used to create what is called a Network Tasking Order (NTO) [11] in order to supplement prediction tools thus allowing their predictions more viable. Previous research discovered that such information could enable more optimal network outcomes [15]. Past work has also demonstrated the ability of an agent-based framework to optimize network behavior using long-term, mid-term, and short-term estimates of network behavior [25].

1.2 Goal

This research concentrates on three main goals. The first is to expand the use of prediction tools, i.e. Kalman filter [18] and past networking optimization algorithms [15]. Secondly, employ these prediction tools in an emulated airborne network. Finally, use these prediction tools to maximize the use of limited bandwidth available in airborne networks. This thesis concentrates on the use of the Unscented Kalman Filter (UKF) [7] to predict queue sizes of routers placed at different points in an emulated airborne network environment. Making use of these predictions is the responsibility of a centralized intelligent agent to route network traffic effectively throughout the network to increase overall network throughput. The Kalman filter uses current router queue size to predict future queue size. If the queue size is predicted to be greater than a predetermined threshold, network congestion is noted and passed along to the network controller. At this time, the network control algorithm determines optimal network paths and reroute network traffic through non-congested routes to ensure successful transmission of the traffic.

1.3 Research Contributions

This thesis combines several prior research efforts to use the UKF to predict network congestion and through the use of a centralized network controller, maximize throughput in a emulated airborne network. Past research developed the UKF and other Kalman filters, and network control algorithms to control network flows. Even though these algorithms have been integrated into simulated and emulated static network environments, the Kalman filters and control algorithms have not been integrated into or developed for use on actual network components or in an emulated airborne network.

This research integrates the UKF into an emulated airborne network and determines if it can be used to obtain and predict queue size in routers placed throughout. If integration is successful, these predictions determine if network congestion is present at any point in the network. If network congestion is predicted, network nodes provide updates to a centralized network controller. The network controller makes routing decisions based on updates provided by the network nodes. The routing decisions determine the best network route for each node that provides updates. Routing information is based on the latest update received. If network nodes fail to provide updates, routing information may be outdated. The new routing information is made available, upon request, to all network nodes. The routing information contains individualized network route changes to minimize network congestion, reduce latency, and improve the overall network throughput.

1.4 Conclusion

The dynamic nature found in airborne networks requires a highly specialized network controller. A queue size prediction scheme in conjunction with a network control algorithm could fill this need and enhance overall network effectiveness which is an important factor for today's airborne environments. Insight into the challenges faced by these types of networks and how to mitigate the challenges is the crux of successful information gathering in airborne environments.

The following is an overview of the chapters discussed in this thesis:

II - Provides necessary background information to demonstrate the research that has already taken place in the area. The literature review also provides an introduction into the terminology and concepts referenced in this document.

III - Provides an overview of the methodology to test the concepts described herein.

IV - Presents the results and an analysis of the data gathered from the experiments.

V - Overall conclusion and general remarks for future work in this area.

II. Literature Review

This chapter discusses research that was previously accomplished in the areas of network prediction tools and intelligent agents in corporate, military and airborne networks. This chapter provides the necessary background to define the problem and discuss current technologies currently in practice today. It is presented in a top-down approach starting with MANETs, replication of a dynamic airborne environment, network prediction tools and finally network controllers. This chapter also explains the Kalman filter, the algorithm used for network prediction, used in conjunction with a network controller to improve overall network efficiency in a MANET by forecasting when links are becoming congested.

2.1 Mobile Ad-hoc Networks

A MANET is a collection of mobile nodes that generate a network automatically by creating several short-lived radio frequency links based on node proximity and mutual agreement [20]. This research concentrates on an airborne MANET, more specifically, a collection of airborne platforms in use by the U.S. Military. A MANET relies on the use of wireless technologies to perform communication between two+ airborne devices, between airborne platforms and a ground station, or between airborne platforms and satellites. As with all wireless technologies, the data rate between two devices is dynamic in nature due to distance between the devices. When the distance between two devices decreases, the link speed increases and conversely, the link speed decreases as the distance between the two devices increase. Each node within a MANET comes equipped with wireless receivers and transmitters. Antennas on a device may be omni-directional, highly directional, or possibly steerable [6].

The bandwidth between the two devices is directly related to the distance between them. As distance increases, the data rate between the two devices decreases until the distance is too great to support communication at which time, the communication link drops completely. If the communication link between two devices drops, a third node, that maintains communication with both devices, may act as an intermediate node and route network traffic between the two devices. Every node in the network participates in a protocol that facilitates the discovery of multi-hop paths throughout the network to any other node [17].

Every node in a MANET is highly mobile and independent of other nodes, therefore, the MANET is highly dynamic and ever evolving where link speeds among nodes change frequently. This type of network requires routing protocols that are just as highly dynamic as the MANET itself. Thus the primary challenge of a MANET is capturing the current network state to properly route traffic throughout the network. Conventional networks rely on link-state or distance-vector algorithms to maintain the correct routing information. Using either one of these algorithms presents issues as well. First, sending periodic updates throughout the network requires using an already limited bandwidth. Second, with the highly dynamic nature of a MANET, the network configuration may change too quickly for routing protocols to maintain up-to-date network information [19].

2.2 Airborne Networks

In today's world, airborne networks are extremely important to gather intelligence data and transmit that data to users on the ground so those users have the information they need to make informed decisions. In an article by B. Cheng [10], he discusses traditional tactical airborne networks (shown in Figure 2.2) which he describes as having the following issues.

- **Long transmission ranges** - Typical ranges vary from 100-300 nautical miles [10].
- **Low data rates** - Tactical airborne networks speed range from 250Kbps - 2Mbps throughput [10].
- **Dynamic data rates** - Data rates in tactical airborne networks can vary data rate and transmit power on a per flow basis. Even between a pair of nodes, packets are sent at different data rates and transmission power [10].
- **Differing multicast and unicast rates** - Multicast and unicast data rates can vary depending on the ability of the node to adjust transmit data rates on a per flow basis [10].
- **Large range in relative speed** - Due to the distances between aircraft, the speed of the aircraft, and direction of travel, the time an aircraft takes to traverse the transmission region can vary between 3 - 72 minutes [10].
- **Periodic and sporadic mobility patterns** - Some aircraft have highly regular flying patterns while other aircraft may have to engage targets outside communication range making them sporadic [10].
- **Periodic and bursty traffic** - Again, some aircraft may have regular network traffic while other aircraft may have short bursts of traffic [10].

2.2.1 Ad-hoc On-demand Distance Vector Protocol.

The Ad hoc On-demand Distance Vector (AODV) protocol is the preferred protocol for use in a MANET. Several factors are appealing for use in these highly dynamic networks. AODV requires low network utilization, processing power, and memory overhead compared to other networking protocols. AODV is able to adapt

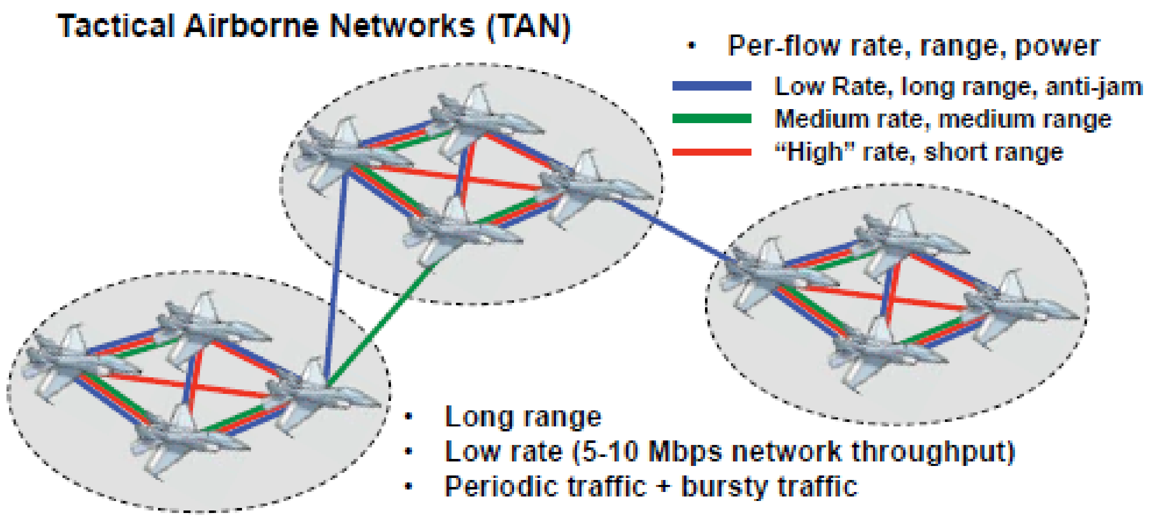

Tactical Airborne Networks (TAN)
• Per-flow rate, range, power
Low Rate, long range, anti-jam
Medium rate, medium range
"High" rate, short range
• Long range
• Low rate (5-10 Mbps network throughput)
• Periodic traffic + bursty traffic

Figure 2.2. Tactical Airborne Network Charararacteristics [10]

quickly to dynamic link conditions and determine unicast network routes within a MANET, unlike static network protocols [12]. AODV requires only two addresses to create a route from one node to another, the designation, and next hop. These features make AODV the preferred network protocol for MANETs [9].

When it comes to MANETs, AODV has some disadvantages also. First, since AODV uses a portion of the network bandwidth to learn routing information, this network traffic increases the overhead of an already taxed network infrastructure. Secondly, AODV is not optimal for a MANET because of the highly dynamic nature of the network. This dynamic quality leads to network routing information to become out-of-date quickly, leading to incorrect routing tables, requiring routing updates to be made more often [19]. The disadvantages listed above are not limited to AODV, but rather, they are inherent problems with any network protocol running on this type of network.

2.2.2 MANET Extension of Open Shortest Path First.

Open Shortest Path First (OSPF) is a good solution for static networks where link states are stable. In situations where link states are highly dynamic such as airborne networks, OSPF is not the best routing protocol to use. Scaling issues regarding the flooding protocol operation, inability of the designated router election protocol to converge in all scenarios and the large number of adjacencies when using a point-to-multipoint interface type make OSPF unviable for use in dynamic networks [23]. According to a study done in 2009 by Z. Wang and J. Crowcroft, "In a dynamic network environment under heavy traffic load, shortest-path routing algorithms, particularly those that attempt to adapt to traffic changes, frequently exhibit oscillatory behaviors and cause performance degradation" [30]. The limitations of OSPF are well known which led to the creation of an OSPF extension in 2009 to support networks deployed in MANET configurations. More specifically, this extension supports broadcast-capable, multi-hop wireless networks in which the host and other network devices are mobile [23].

The OSPF extension has several factors that make it appealing for MANET configurations. First, the modification of hello packets using OSPF link-local signaling and it serves two purposes: to provide neighbors with two hop neighbor information as well as differential hellos that are sent more frequently without a significant increase in overhead, in order to respond more quickly to topology changes. In MANET configurations, each router must advertise some or all of its MANET neighbors as point-to-point links in its router link state advertisement. To reduce the amount of overhead, the choice of which neighbors to advertise to is flexible, directly reducing the amount of topology information that is passed across the network.

2.2.3 Optimized Link State Routing.

Optimized Link State Routing (OLSR) protocol is a proactive, table driven routing protocol designed specifically for use in MANETs [13]. OLSR incorporates the use of Multi-Point Relays (MPRs) which are nodes in MANETS that relay messages between nodes. Besides relaying messages between nodes, the main role of the MPRs is routing traffic and selecting the proper route for traffic flowing between source and destination nodes. OLSR maintains routes to every destination in the network requiring the use of memory. Also, the job of gathering and maintaining all of the routes is computationally taxing on the system, on the order of $\mathcal{O}(n^2)$ [13].

In OLSR, the node first sends hello packets and selects the multi-point relay nodes that relay a node for flooding. The node then floods topology control messages through the multi-point relay nodes and constructs an overall routing table of the network. The OLSR protocol maintains connectivity information on the latest two hop nodes and the multi-point relay node by continuously flooding hello packets [14]. The continuous flooding of hello packets throughout the network decreases the amount of bandwidth for data that requires transmission especially in networks with low bandwidth limitations as in the case of airborne networks.

2.2.4 MikroTik Routing Protocols.

MikroTik router OS includes several protocols that can be utilized, two of which are discussed. The first is Mesh Made Easy (MME). According to the MikroTik user's manual, "MME is a MikroTik routing protocol suited for IP level routing in wireless mesh networks [4]." MME is based on ideas from Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) routing protocol for multi-hop ad-hoc mesh networks [1]. As explained in the MME user's manual, "MME works by periodically broadcasting Transmission Control Protocol (TCP) packets so called originator messages." MME does not maintain network topology information and therefore is not able to calculate a routing table, nor does need to. MME keeps track of lost packets by scanning originator message packets received and their corresponding sequence numbers. If sequence numbers are missing, MME is able to determine the number of lost packets. By using this information and using the combinations of originators and single-hop neighbors, MME is able to find the best route to a particular destination instead of using a routing table [4]. The MikroTik router OS also includes OSPF version two. According to MikroTik, "The OSPF protocol is the link-state protocol that takes care of the routers in the dynamic network structure that can employ different paths to its subnets. It always chooses shortest path to the subnetwork first." [4].

2.3 Airborne Networking Routing Protocols

While studies of MANET routing protocols have increased over the last couple decades, the focus of these studies have been on mesh and ground networks leaving a void in airborne networks. Currently, two different types of routing protocols are in use, static and dynamic routing protocols. Static routing is simply the manual entry of routes into the routing table. The router looks through the routing table to determine the path for the route. As stated, this method requires manual entry of routing information and is not considered a protocol. Static routing has a couple inherent problems: a large number of networks maintain too many routes to manually be input and large delays due to rerouting network traffic due to outages exists. Dynamic routing on the other hand are protocols where software applications actively seek out connections and network routes to destinations. In a dynamic routing protocol, the router learns its directly connected neighbors first and then each router tells its neighbors the routes that they know thus propagating this information throughout the network. After sorting through the list of routes, the protocol determines the best route to each network it has traffic destined to. The sharing of routes gives dynamic routing protocols the ability to adapt to network topology changes or outages.

Along with static and dynamic routing protocols, three major categories of ad-hoc routing protocols exist: reactive, proactive and hybrid. Reactive protocols determine routes on-demand which limits the amount of traffic transmitted across the network. Since network paths are determined on an as needed basis, this eliminates the constant updating of routing tables. Since each route is determined as needed, the route discovery process is on-demand by flooding Route Request (RReq) packets throughout the network [28].

Proactive routing consists of each network node uses their own routing table to establish network connections to other nodes in the network. Each node records discovered destinations and the respective number of hops to the discovered node in the routing table. A sequence number, created by the destination node, is also recorded in the routing table. To ensure accuracy of the routing table, each node broadcasts and modifies their routing table from time to time. Due to the amount of network traffic generated to update routing tables, proactive protocols are appropriate for networks with a limited number of nodes [28].

Hybrid protocols incorporate the merits of proactive as well as reactive routing protocols. A hybrid routing protocol should use a mixture of both proactive and reactive approaches [24].

With the dynamic nature of airborne networks, static routing is not feasible. Inherent problems exist when trying to apply dynamic network protocols designed for static networks into MANETS. Overhead in terms of bandwidth and processing requirements limits the use of these protocols in airborne networks or a MANET. Flooding packets throughout these networks cause unnecessary overhead on a link's capacity, potentially creating many transient routing loops. These problems limit scalability of the routing protocols to such large routing areas and limit the protocol's use to static networks.

2.4 Network Optimization

Network state prediction and network control play important roles in network optimization. Past research have shown that the possibility exists to implement automatic network controllers to adjust routes throughout the network for the purpose of congestion control and route optimization [15]. Prediction schemes predict the future network states and pass this information to the network controller for route optimization. Using a prediction model allows the network controller to make accurate decisions and mitigate network congestion and utilize limited network resources more efficiently. As with any prediction tool, the prediction interval should be large enough to offset any delays caused by network congestion but be small enough to make accurate predictions. Queue size at any given time depends on the bandwidth of the link and the amount of traffic on that link at any given time. At one point in time, the network link could be empty whereas the next second, the link could be completely saturated. The dynamic nature of network traffic equates to a ever-changing router queue size in a nonlinear system, especially in an airborne network where link speeds are constantly changing.

In current MANET routing schemes, routing changes use a significant amount of bandwidth to make routing changes throughout the network. A prediction tool, that predicts when and where congestion occurs, can be utilized to reduce network overhead and allow for efficient use of network bandwidth.

Prediction tools have the flexibility to predict at different intervals into the future. The prediction tool must predict far into the future so predictions of network traffic maintain accuracy. Past research has shown that the prediction accuracy decreases as the prediction interval increases [15]. Automatic network controls making decisions based on erroneous predictions, can deteriorate overall network performance [27].

2.4.1 Kalman Filter.

The idea of the Kalman filter was developed by Dr. Rudolph E. Kalman in 1960 to aid in spacecraft navigation. The Kalman filter is a least mean square algorithm that is capable of estimating an unknown state of dynamic processes. The filter is a set of mathematical equations used to estimate the future state of a dynamic system by using the previous estimated state mathematically combined with the current state. A Kalman filter is a recursive solution for a linear filtering problem. The Kalman filter involves determining a predicted value of a system when given a set of noisy measurements from that system. The Kalman filter supports predictions of past, present and future states of a given system.

The Kalman Filter has two main states, predict and update, as shown in Figure 2.3. Figure 2.3 shows how the Kalman filter estimates the next state using only the previously estimated value and the current value, therefore, no other predictions or state values are needed. The predict phase, called the *a priori* state, is the current time step estimate. The update phase, known as the *a priori* prediction, is the current state value. The *a priori* prediction is captured in the refined state estimate called the *a posteriori* state estimate [15].

Maybeck modeled the Kalman filter from:

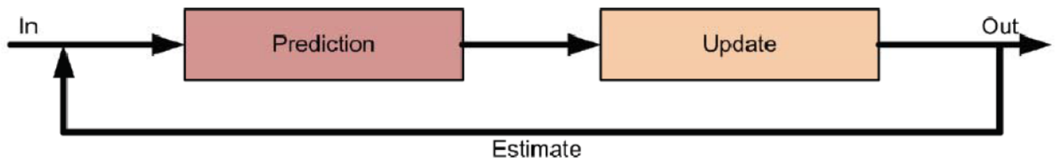


Figure 2.3. Kalman Filter - Predict Update Process [22]

$$x_k = F_k x_{k-1} + B_k u_k + w_k \text{ [21]}, \tag{2.1}$$

where

- F_k is the state transition model and applied to the previous state of the model,
- B_k is the control-input model and applied to the control vector u_k,
- w_k is the process noise from a zero mean normal distribution with covariance Q_k in:

$$w_k \approx N(0, Q_c) \text{ [21]}. \tag{2.2}$$

At time k a measurement z_k of the true state x_k is according to

$$z_k = H_k + v_k \text{ [21]}, \tag{2.3}$$

where H_k is the observation model that maps the true state space into the observed space and v_k is the observation noise which is assumed to be zero mean Gaussian white noise and covariance R_k in:

$$v_k \approx N(0, R_k) \text{ [21]}, \tag{2.4}$$

The state of the Kalman filter is represented by two variables:

1. $\hat{x}_{k|k}$ is *a posteriori* state estimate at time k given observations up to and including at time k [21], and
2. $P_{k|k}$ is *a posteriori* error covariance matrix, which is a measure of estimated accuracy of the state estimate [21].

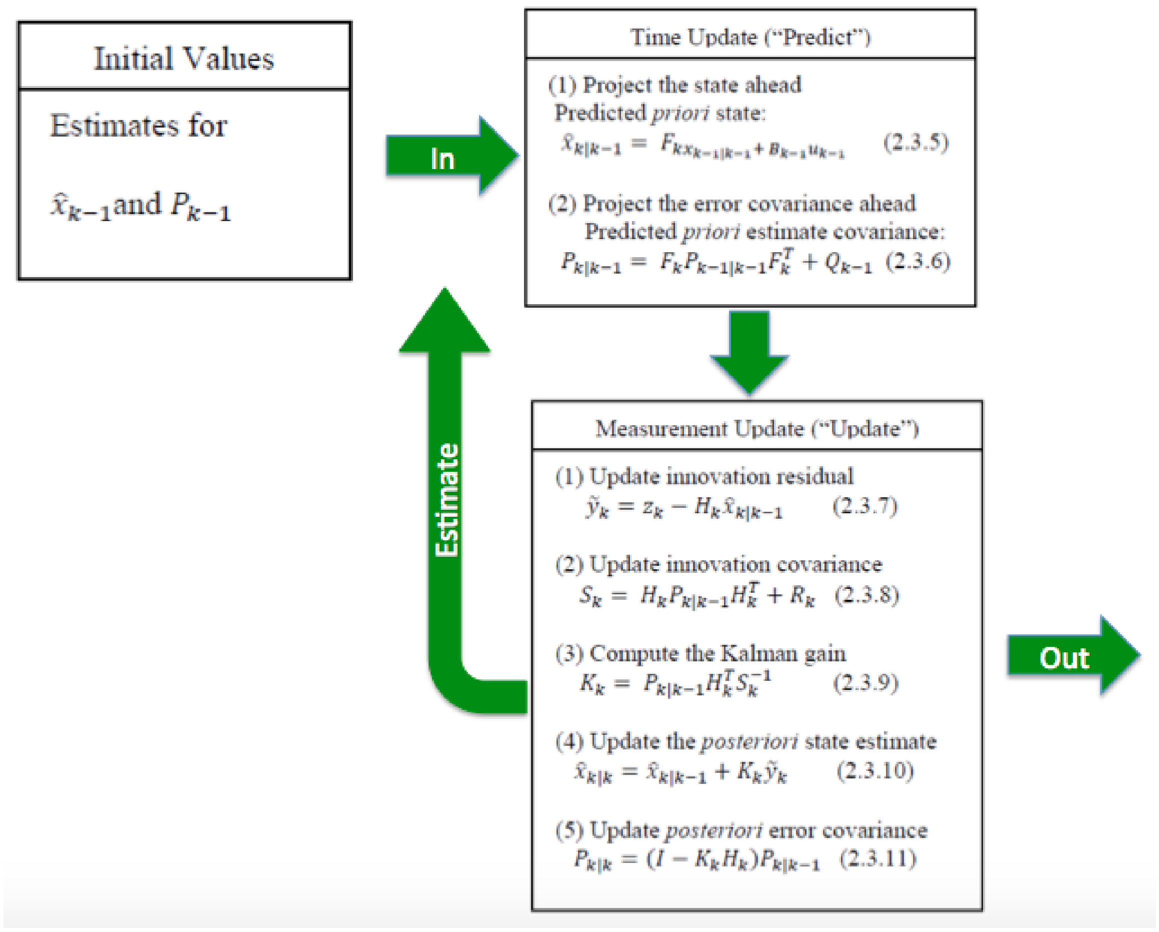

Initial Values
Estimates for
$\hat{x}_{k-1}$ and P_{k-1}
In
Time Update ("Predict")
(1) Project the state ahead
Predicted *priori* state:
$\hat{x}_{k|k-1} = F_k x_{k-1|k-1} + B_{k-1} u_{k-1}$ (2.3.5)
(2) Project the error covariance ahead
Predicted *priori* estimate covariance:
$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{k-1}$ (2.3.6)
Estimate
Measurement Update ("Update")
(1) Update innovation residual
$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$ (2.3.7)
(2) Update innovation covariance
$S_k = H_k P_{k|k-1} H_k^T + R_k$ (2.3.8)
(3) Compute the Kalman gain
$K_k = P_{k|k-1} H_k^T S_k^{-1}$ (2.3.9)
(4) Update the *posteriori* state estimate
$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$ (2.3.10)
(5) Update *posteriori* error covariance
$P_{k|k} = (I - K_k H_k) P_{k|k-1}$ (2.3.11)
Out

Figure 2.4. Overview Chart of Kalman Filter Process [15]

The function $\hat{x}_{n|m}$ represents the estimate of x at time n given observations up to, and including time m [21].

The Kalman filter uses two different mathematical update methods: time update and measurement update. The time update equation uses error covariance to attempt to predict the future state and passes the information to the measurement update. The measurement update then passes the new estimate to the process using the data as well as back into the time update which uses this new prediction to predict the next value [21]. The summary of this process is shown in Figure 2.4.

2.4.2 Unscented Kalman Filter.

Past research has tested a couple types of Kalman filters (Extended [15] and Unscented [7]). The most common approach is the use of the Extended Kalman Filter (EKF) which simply linearizes all nonlinear models to make use of the traditional linear Kalman filter [16]. According to Julier, "Although the EKF is a widely used filtering strategy, over thirty years of experience with it has led to a general consensus within the tracking and control community that it is difficult to implement, difficult to tune, and only reliable for systems which are almost linear on the time scale of the update intervals" [16]. Two main drawbacks by the EKF are: linearization can cause the filter to become highly unstable when there is a violation of the assumptions of local linearity and the derivation of the Jacobean matrices are nontrivial in most applications and make implementation difficult [16]. For these reasons, the Unscented Kalman Filter (UKF) was developed. The UKF is a novel method for calculating the statistics of a random variable which undergoes a nonlinear transformation. The UKK is better suited than linearization for filter applications.

UKF was recently researched by Alqahtani [7]. Alqahtani's research consisted of tests on two different network models. The first model consisted on a two router design shown in Figure 2.5. Alqahtani used the first model to test three different types of filters and two prediction periods: a basic filter, EKF, and UKF and 1 second, and 5 seconds prediction times. According to Alqahtani "The UKF gave the best prediction performance compared with other filters [7].

The second model was a four router design shown in Figure 2.6. For the second model, Alqahtani used the filter queue size state prediction model result and fed it into the queue controller. The queue controller was used, and according to Alqahtani, the number of packets dropped was reduced and improved network performance [7].

2.5 Network Forecasting

The Kalman filter is used to estimate the future queue size to aid in the process of forecasting network congestion. If the Kalman filter correctly predicts a router's queue size and in-turn network congestion. The correctly predicted network congestion passed along to a network controller what aids in relief of this congestion by sending

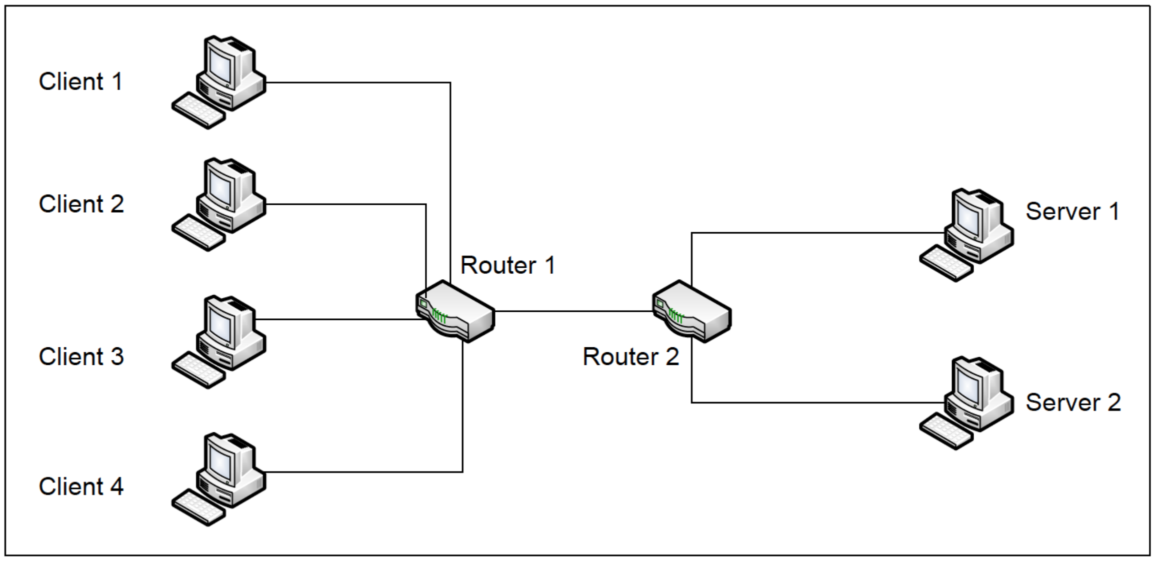


Figure 2.5. Alqahtani Two Router Design [7]

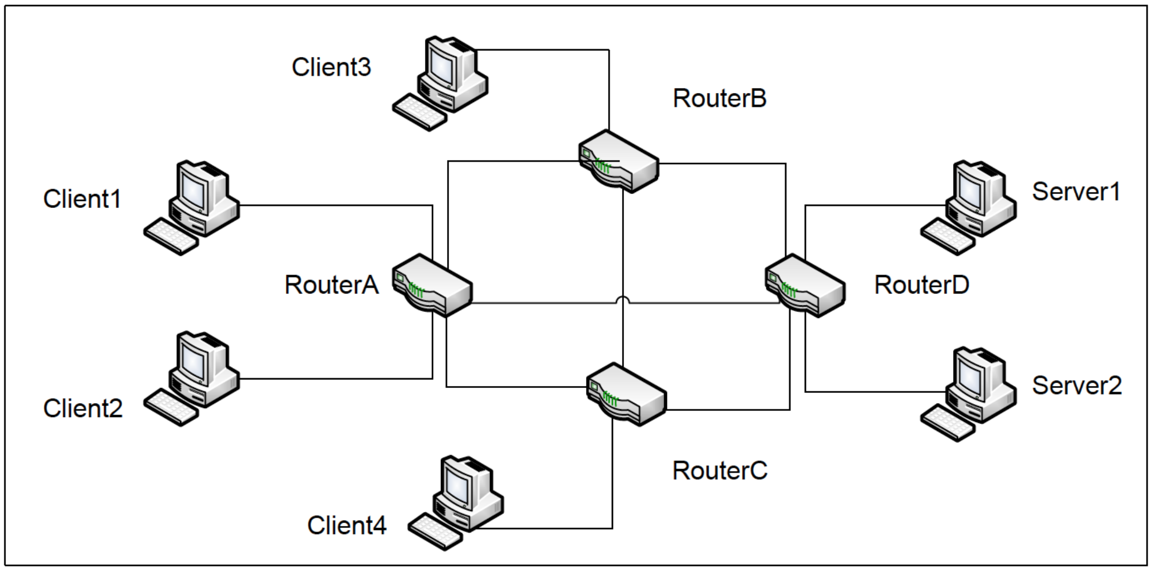


Figure 2.6. Alqahtani Four Router Design [7]

data through different routes in the network. Haught performed several simulations testing the concept of the Kalman Filter to predict network queue size [15]. The simulation network consisted of 14 nodes with five different Kalman filters placed in key locations throughout the network as shown in Figure 2.7.

All router queue's had a maximum queue size of 1000 packets. Network traffic for Haught's research consisted of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets. The Kalman filters accurately predicted the queue size every second for 500 seconds of simulation time. The Kalman filter's accuracy declined if predicting too far into the future. Haught discovered that the predictions were most accurate when predicted second into the future (see Figure 2.8).

Haught performed several different simulations using the network described above and adding a Dynamic Routing Queue Controller (DRQC) that acted as a network controller [15]. This network controller managed the network traffic in the network by

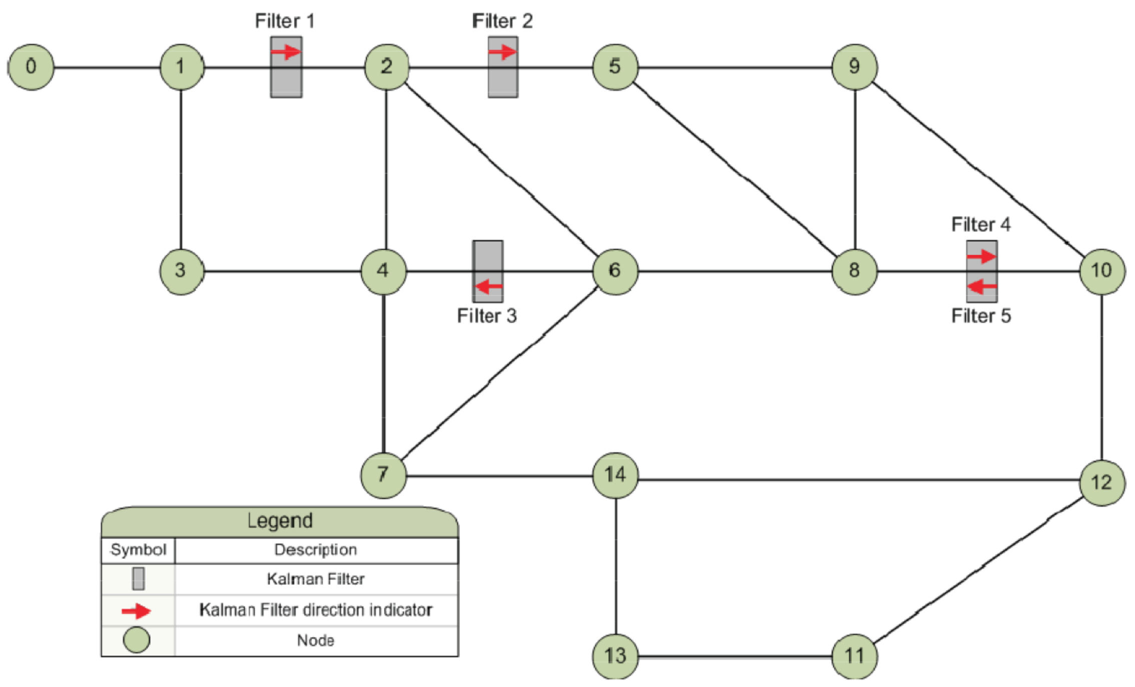


Figure 2.7. Haught's Simulated Network Diagram for Kalman Filter Validation [15]

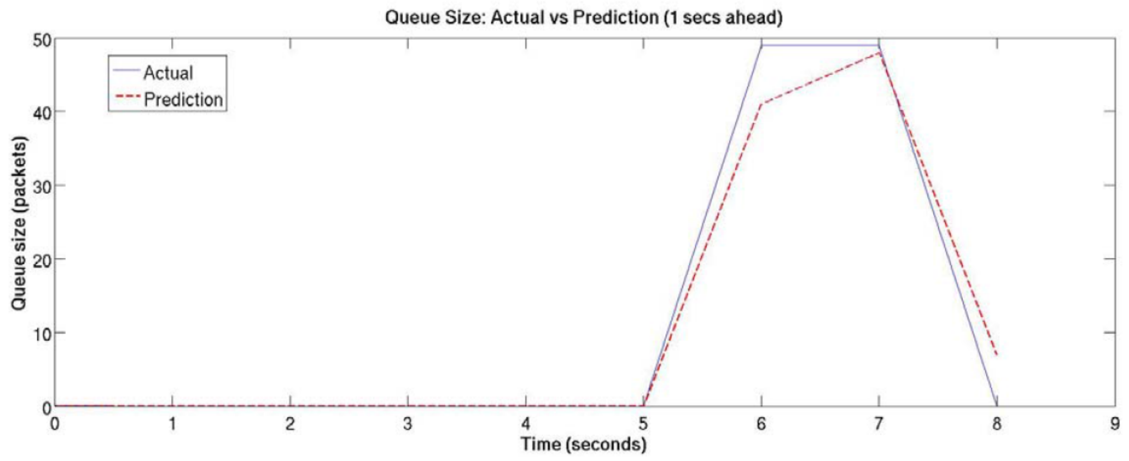


Figure 2.8. Haught's Implementation of the Kalman Filter with 1 Second Predictions [15]

adjusting the routes throughout the network based on network congestion. Haught used the Kalman filter to predict the router queue size. When the router queue size prediction exceeded a threshold, it was noted as network congestion. Kalman filters were placed at the strategic locations throughout the network. Using the DRQC to control the network flow, overall network congestion was reduced and network throughput increased. Haught also discovered that using the Kalman filter to predict router queue size, increased the overall DRQC performance [15].

2.5.1 Multi-commodity Flow.

Current networks implement a shortest path algorithm to determine the route for network traffic. The shortest path algorithm does not achieve the desired results in environments where the nodes are constantly moving as in the case of airborne networks. Wang did some studies in this field and discovered that in a dynamic network with higher than average network traffic, the shortest path algorithms have performance issues due to oscillatory behaviors, especially in networks that that try and adapt to traffic changes [30].

Multi-commodity flow paradigm is a network flow paradigm that maximizes the amount of network traffic that is able to travel across a network from multiple sources to their respective destinations where each link is subject to differing capacity constraints. A multi-commodity flow consists of a set of ordered pairs of vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ where each pair represents a commodity with a source s_i and a target t_i. For each commodity (s_i, k_i) the commodity specifies a non-negative demand d_i [5]. The objective of the multi-commodity flow is to maximize the amount of traffic flow traveling from multiple sources to their corresponding destinations simultaneously as shown in Figure 2.9. Figure 2.9 demonstrates s_1 sending traffic to t_2 while s_2 is sending traffic to t_1. Figure 2.10 presents a solution to the multi-commodity flow problem using fractional flows. This solution allows 100% of traffic to travel from nodes s_1 and s_2 to their respective destinations.

Betances implemented a breadth-first search algorithm to maximize the amount of flow traveling from various sources to their corresponding destinations all subject on the individual link capacity constraints [8]. The algorithm computes an approximation using fractional flows in order to minimize the computing resources required.

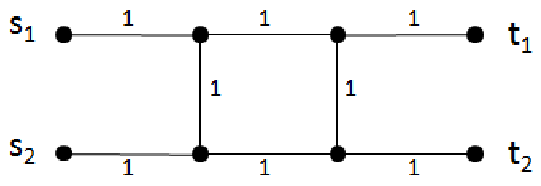


Figure 2.9. Multicommodity Flow Problem $d_1 = d_2 = 1$ [5]

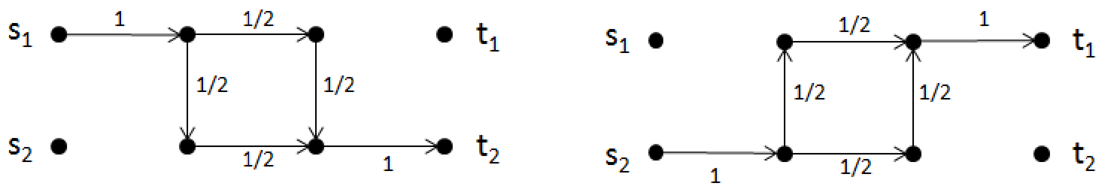


Figure 2.10. Multicommodity Flow Solution to Problem in Figure 2.9 [5]

The algorithm works as follows. First, the algorithm initialized all edge lengths of the topology. The initial length is small and is the base to exponentially increment the lengths as flows congest edges [8]. Next, the algorithm cycles through commodities lengthening edges based on a feasible shortest path routing. Once the path is known, an increase is made to the edge lengths $l(e)$ associated with this path to avoid over utilization of a particular edge [8]. The exit condition of the algorithm is when the sum of all edge weights $(D(l))$ is greater than one. A constraint is set on the maximum error by the value of ϵ, the constant used to increment the edge lengths $l(e)$ [8]. Figure 2.11 illustrates the pseudocode of the algorithm.

2.5.2 Dynamic Route Queue Controller.

Haught developed the idea of the DRQC, a centralized network controller that reroutes network flows based on flow priority, queue predictions, and network congestion [15]. The implementation of the DRQC uses the Kalman filter to predict the state of the network and optimize the network flow based on that prediction. The Kalman filter predicts the queue size of routers placed throughout the network and

```
Initialize $l(e) = \delta/\mu(e)\ \forall\ e,\ x \equiv 0$.
while $D(l) < 1$
    for j = 1 to k do
        $d'_j \leftarrow d_j$
        while $D(l) < 1$ and $d'_j > 0$
            P ← shortest path in $P_j$ using l
            u ← min $\{d'_j,\ min_{e\in P}\mu(e)\}$
            $d'_j \leftarrow d_j - u$
            $x(P) \leftarrow x(P) + u$
            $\forall\ e \in P,\ l(e) \leftarrow l(e)(1 + \frac{\varepsilon u}{\mu(e)})$
        end while
end while
Return (x;l).
```

Figure 2.11. Multi-commodity Flow Algorithm [8]

when the queue size prediction is above a predetermined threshold, the controller determines the network is congested. When the controller detects network congestion, it adjusts the network to reduce or eliminate the congestion. The DRQC is a centralized network controller to maintain the network by reading data points throughout the network. Another feature of the DRQC was to allow for prioritized network flows thus allowing packets to contain differing levels of priority. This priority leveling allows higher priority traffic to utilize network bandwidth when the network is congested. The main purpose behind the DRQC was to show that controller can use the Kalman filter predictions to optimize a network.

The DRQC has five main features:

1. **Priority routing** - changes network flows with respect to the priority
2. **Priority flow control** - starting and stopping flows according to priority
3. **Prediction detection** - controller's ability to utilize Kalman filter queue size predictions
4. **Dynamically split flows** - allows that controller to split flows
5. **Flow reactivation** - restarting stopped flows

The DRQC makes decisions for the network based on the actual and predicted queue sizes of Kalman filters placed throughout the network. Queue sizes are received by the DRQC, the DRQC makes network route decisions based on the queue sizes received. This process is shown in Figure 2.12. The DRQC makes no changes to the network until the network congestion is detected. Once the controller detects congestion, the DRQC makes changes to routing tables and adjusts network flows to relieve the congestion. Haught determined there is network congestion based on the queue size, actual or predicted. Penttinen concluded that once the actual or predicted queue was 50% full, the next time step result in a full queue or the queue would have a long enough delay to cause network congestion or packet loss [26]. Link performance equation:

$$link\ performance = \frac{(packet\ size)}{(\frac{(packet\ size)}{(original\ capacity)})(number\ of\ packets\ in\ queue)}[26]. \tag{2.5}$$

The equation was created by Penttinen for use in networks to determine overall link performance.

DRQC implemented an approach named *exponential backoff*, currently integrated into carrier sense multiple access with collision avoidance (CSMA/CA) and carrier sense multiple access with collision detection (CSMA/CD) networks to retransmit frames. DRQC implemented exponential backoff to solve the problem of a stagnant queue size due to the links being at full capacity. Exponential back-off reduces the current sending rate of the network flows by half allowing queue sizes to reduce to zero for the next time iteration if there is no introduction of other network flows [15].

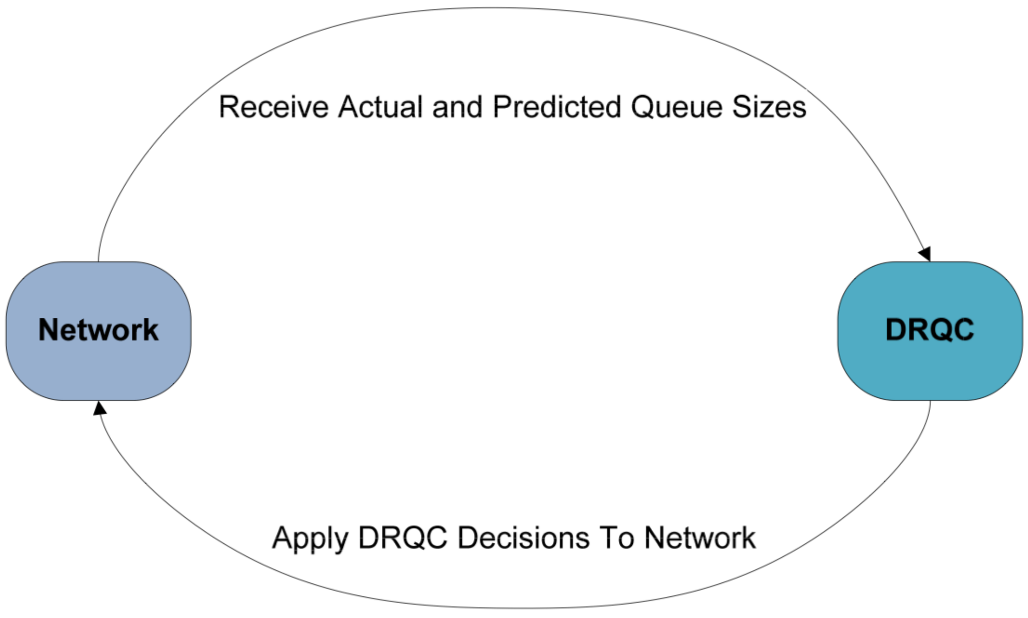

Receive Actual and Predicted Queue Sizes
Network
DRQC
Apply DRQC Decisions To Network

Figure 2.12. State Diagram of DRQC [15]

DRQC's second approach of network congestion handling is execution of DRQC's rerouting process resulting in stopped, started or rerouting of network flows to eliminate network congestion. With the use of Kalman filters, the DRQC is aware of actual and predicted queue sizes and where in the network the congestion occurs. Using this information, the DRQC routes network flows in order of priority [15].

2.6 Summary

This chapter presents the fundamental research and concepts in the areas of MANETs, network prediction tools, and network forecasting. Additionally, Chapter 2 introduced the idea of Kalman filter use to predict router queue size in the aid of network congestion prediction. The next Chapter discusses how an airborne network is emulated using only static network components in order to gather the information needed to optimize QoS in highly dynamic network environments. All of the accomplishments in this area show the application of these techniques to simulated static networks. This research takes these accomplishments, applies them in an emulated airborne network and provides QoS and throughput needed for military airborne networks.

III. Methodology

3.1 Introduction

This chapter describes the methodology used to design and implement the Kalman filter and associated network controller to improve network performance when compared to traditional MANET routing protocols. This chapter outlines the goals and hypothesis of this research, details the problem, describes the environment used, and illustrates the procedures by which the results of procedures are evaluated. Section 3.5 describes the process to create an emulated dynamic airborne network. Section 3.6 is the validation and evaluation of Kalman filters. Once validated, the network controller can use the Kalman filter predictions to enhance the airborne network QoS and throughput. Section 3.7 describes the implementation of the network controller and how the controller uses predicted queue size values from the Kalman filter to react to ongoing network congestion.

3.2 Research Objectives

The objective of this research is to use Kalman filters to accurately predict router queue size in networks where link speeds constantly change and use the predictions to enable a network controller to reroute network traffic through routes with higher bandwidth. The controller computes the optimum route based on network traffic and network topology at the time of the request. Furthermore, the controller executes the algorithm when the predicted router queue size reaches a predetermined threshold. The network updates, route requests, updated routing information, and regular network traffic use the same communication links to ensure realism of network conditions.

3.3 Research Hypothesis

This research evaluates the following four hypotheses.

- Can dynamic airborne link speeds be emulated using static server hardware and network nodes.
- Can Kalman filters accurately predict router queue size in highly dynamic airborne networks.
- Can the centralized network controller receive router updates from network routers when Kalman filters predict network congestion. From routing updates, the network controller is able to determine optimal routing information and communicate this information to requesting routers.
- The number of dropped packets is lower with the application of the network controller compared to MME and OSPF routing protocols.

3.4 Assumptions / Limitations

There are several assumptions to address and several limitations discovered during the course of this research. The assumptions are as follows. The four primary nodes that transmit data across links with variable link speeds are capable of determining link speeds to all secondary nodes. The four primary nodes are capable of capturing current queue size at one second intervals. At all times, at least one connection between the primary and secondary nodes is capable of transmitting all data, i.e., multiple links are not required to ensure successful transmission. This research assumes all platforms have a minimum transmission speed of 64 kilobits per second (kbps) and a maximum transmission speed of 1 Megabits per second (Mbps). The minimum connection speed in this research is based on the minimum speed allowable by the hardware. All other speeds for this research are based off of the minimum to test variable connection speeds and may not be link speeds seen using actual hardware. All network traffic used for this research is UDP which is representative of the streaming video which is usually transmitted by smaller airborne platforms. TCP packets would attempt to retransmit during normal operation causing more packets to be sent causing an inaccuracy in the test results. UDP allows for best effort delivery, i.e., if a UDP packets do not gets sent to the destination, the sender does not attempt to retransmit the packet.

The limitations are as follows. The use of only one gateway is available at any given time, reducing the capability of using a multi-commodity flow algorithm to route data across several links at any given time. The MikroTik router, upon changing link speeds to a given node, drops all queued packets prior to the link speed change instead of transmitting to the destination. This limitation was discovered during initial phases of testing with small number of packets being sent and is an inherent issue with the MikroTik router OS. This is an issue during all tests so DNUC, MME and OSPF are all affected. If all protocols are equally affected, the results of all three protocols show this issue equally. Protocols available on MikroTik router OS and designed algorithms are the only available protocols for this research. Only ten routers were made available for use during this research, this limits the overall size of the network. Link speeds available when route change request is issued drive routing decisions. 64 kbps is the lower limit of the MikroTik Router OS queue, therefore, link speeds between two nodes never degrade beyond 64 kbps.

3.5 Airborne Network

Previous research in this area relied on network simulators to provide the backbone on which to conduct individual research. To further this area of research, migration from a simulation based network to a combination of actual hardware and virtual workstations is required. This migration was necessary to prove concepts developed in simulators, work as effectively in actual hardware. To emulate the airborne environment, virtualization of all individual nodes takes place on a Windows-based server running VMware Workstation 10. Virtual workstations included a combination of Windows Enterprise 7 workstation for network device administration and Ubuntu 12.04 LTS (Precise Pangolin) to handle all data transmission. MikroTik router OS accomplishes all Network routing. As previously discussed, a single server running VMware Workstation hosts all network devices.

3.5.1 Network Topology.

Due to limitations of hardware used and since the purpose of this research is to test concepts that rely on variable transmission speeds, data rates are for proof of concept only and may not represent actual data rates. Betances completed similar research using a simulator. He discovered that some military radios support a maximum rate of 10 Mbps while other support 8 Mbps [8]. He also discovered that as distance between two devices increases, the amount of bandwidth decreases. For his simulations, he used 8 down to 1 Mpbs [8]. For this research, the maximum transmission speed is 1 Mbps, decreasing every minute until the minimum rate of 64 kbps is achieved, stepping down in increments of 128 kbps, 64 kbps and 32 kbps. The rates for this research were chosen as to decrease the amount of traffic that had to be generated to test the Kalman filter. For these reasons, Table 3.1 lists the transmission speeds between each device upon startup of the device, but may not be the actual speeds at the start of each test. Speeds shown in red imply the link speeds are decreasing while speeds shown in green imply the link speeds are increasing, finally, link speeds listed as a “-” have no direct connection between them. Link speeds between nodes

Table 3.1. Transmission Speed (kbps) Between Nodes

Nodes	1	2	3	4	5	6	7	8
1	∞	-	-	-	64	512	384	-
2	-	∞	-	-	192	256	768	-
3	-	-	∞	-	1000	192	256	-
4	-	-	-	∞	320	640	96	-
5	64	192	1000	320	∞	512	512	1000
6	512	256	192	640	512	∞	512	1000
7	384	768	256	96	512	512	∞	1000
8	-	-	-	-	1000	1000	1000	∞

five through eight have static link speeds. All other link speeds are continuously varied in line with the following progression schedule (in kbps), 64 $\rightarrow$ 96 $\rightarrow$ 128 $\rightarrow$ 192 $\rightarrow$ 256 $\rightarrow$ 320 $\rightarrow$ 384 $\rightarrow$ 512 $\rightarrow$ 640 $\rightarrow$ 768 $\rightarrow$ 1000 $\rightarrow$ 768 $\rightarrow$ 640 $\rightarrow$ 512 $\rightarrow$ 384 $\rightarrow$ 320 $\rightarrow$ 256 $\rightarrow$ 192 $\rightarrow$ 128 $\rightarrow$ 96 $\rightarrow$ 64. The link speed progression allows for constantly changing data rates but may not be what is seen in actual hardware.

Figure 3.13 shows the network setup for this research. Nodes 1-4 are primary nodes and represent small airborne platforms that are unable to transmit network traffic directly to a ground station. Each node has two computers attached, one administrative and one data transmitting computer. The administrative computer handles all Kalman filter predictions and if necessary, rerouting network traffic from it's corresponding node to the secondary nodes. The secondary nodes consist of nodes 5-7. These nodes act as larger airborne nodes that can receive network traffic from smaller nodes and route it directly to a ground node. Node 8 is the ground station in this scenario. The ground station receives all of the network traffic from the four primary nodes. The ground station also acts as the network controller receiving updates from the primary nodes, determining best possible network paths for all primary nodes, and transmits routing information as the primary nodes request changes in their corresponding network routes.

Configuration of all network link speeds is completed using the MikroTik Router

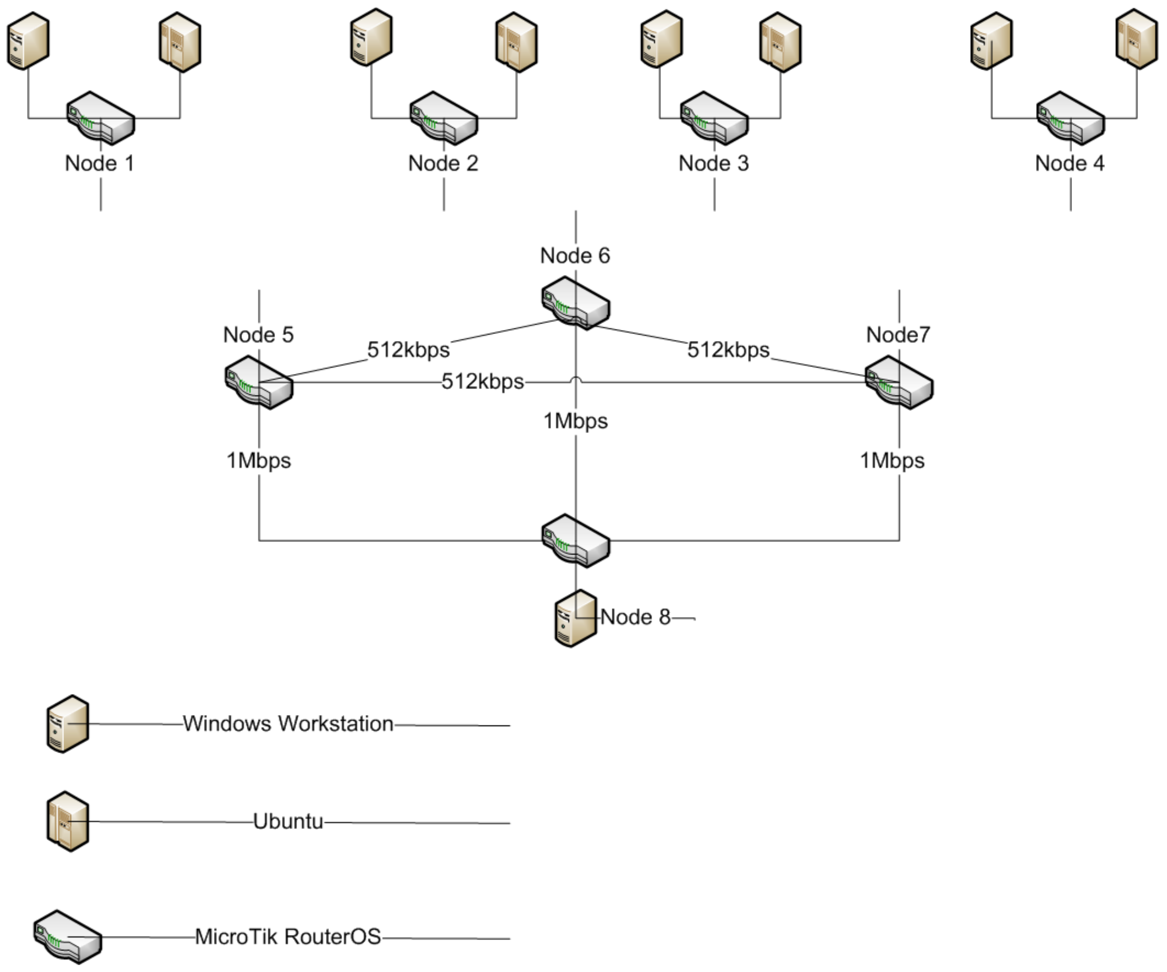


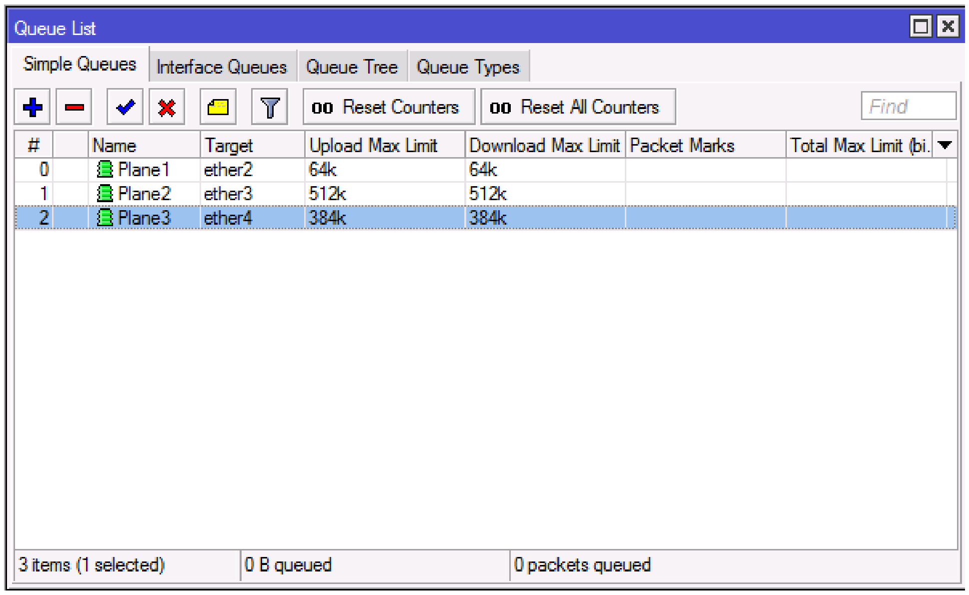
Figure 3.13. Network Design used to all Testing

OS. MikroTik has options available allowing the configuration of queues to specific Internet Protocol (IP) addresses or interfaces of the router itself. Controlling the queue to a specific interface allows for configuring link speeds to specific routers. Controlling link speeds to different routers is the basis for simulating the variable link speeds to different airborne platforms. Figure 3.14 shows the MikroTik configuration of the variable link speeds. The target listed in the figure corresponds to a specific interface on the router itself which directly connects to a secondary router which acts as a different plane in the simulated airspace. All commands to control link speeds can be found on the MikroTik manual web page [4]

The router directly controls each interface using the scheduler function within the

MikroTik router. The scheduler controls each interface and the interface's link speed independently allowing for variable link speeds on each interface. Upon configuration of the queue as shown in Figure 3.14, the scheduler can be configured to change the link speeds at an predetermined interval as shown in Figure 3.15.

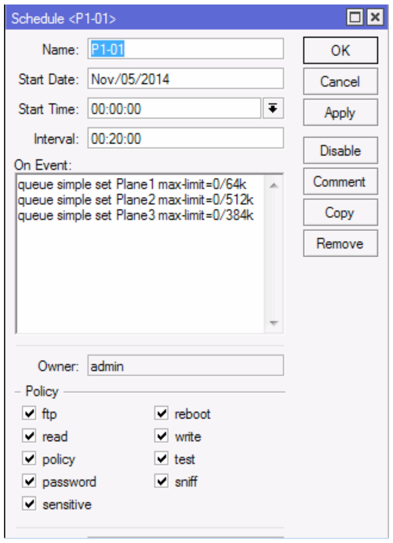
Figure 3.13 shows configuration of the network used to perform the data collection. The four nodes on the top represent smaller aircraft not able to transmit data directly



#		Name	Target	Upload Max Limit	Download Max Limit	Packet Marks	Total Max Limit (bi.
0		Plane1	ether2	64k	64k		
1		Plane2	ether3	512k	512k		
2		Plane3	ether4	384k	384k		

3 items (1 selected) | 0 B queued | 0 packets queued

Figure 3.14. Sample MicroTik's Queue Configuration



Name: P1-01
Start Date: Nov/05/2014
Start Time: 00:00:00
Interval: 00:20:00
On Event:

```
queue simple set Plane1 max-limit=0/64k
queue simple set Plane2 max-limit=0/512k
queue simple set Plane3 max-limit=0/384k
```

Owner: admin
Policy
✓ ftp ✓ reboot
✓ read ✓ write
✓ policy ✓ test
✓ password ✓ sniff
✓ sensitive

Figure 3.15. Sample MicroTik's Scheduler

to a ground node. Therefore, these nodes transmit their data through larger aircraft (represented as the three nodes in the center of the figure) which route the data to the ground node (represented as the node on the bottom of the Figure 3.13). The ground node is the destination node for the primary nodes data transmission. As well as being the destination node for all data, the ground node acts as the Dynamic Network Update Controller (DNUC) provides all necessary routing changes for the primary nodes. The network link speeds are variable between the primary nodes and the secondary nodes according to the schedule above. At any given time, each primary node has one link that is able to support all network traffic from that node. Upon activation of the scheduler, the link speeds change according to the predetermined schedule.

3.5.2 Network Validation.

Success of this research depends on simulating highly dynamic link speeds of an airborne network environment by using static network components. Validation of the variable link speeds is necessary for validation of all experiments. Validation of the dynamic link speeds is conducted using the ping tool. Sending one 500 kilobits (kb) ping, every five seconds while the link speed are set at 1 Mbps, 500 kbps, 256 kbps, and 128 kbps. If the link speeds are configured and the MikroTik router OS is working correctly, the Round Trip Times (RTT) are 0.5 seconds, one second, 2 seconds, and four seconds respectively.

3.6 Kalman Filter

The design of the network controller reacts on state predictions made by the Kalman filter. The Kalman filter used for this research is the UKF which was tested during the research of Alqahtani [7]. To validate the Kalman filter predictions, the future state is compared to the observed state. In previous research, the research was conducted using static networks, run in networks simulators such as Network Simulator version 2 [15]. Limited research in airborne networks was a simulation of the Caspian Sea Scenario described by Betances [8]. The goal of this research is a more realistic airborne network consisting of actual network devices communicating over an actual network. Ensuring the network is more realistic, airborne nodes, links, and network traffic are emulated while the Kalman filter predicts the network state in one second intervals. The predictions are sent to the network controller to make network wide routing decisions.

Validation of the Kalman filter in an airborne network consists of placing a Kalman filter in a network router and sending network traffic through the router. The amount of traffic sent through the router is more than the destination link can support to ensure the router the receive rate of the router is larger than the send rate, i.e., the router queue size increases. As the network traffic is sent through the router, the Kalman filter predicts queue size in the router. Each time the Kalman filter makes a prediction, the prediction is stored along with the measured value and a naïve prediction value. The naïve prediction is the previous measured value. Verification of the Kalman filter consists of comparing the Kalman filter and naïve prediction values against the current state of the queue at the predicted time. Several iterations are captured with the smaller mean difference between the predicted value and the measured value have a better prediction method.

3.7 Dynamic Network Update Controller

Similar to other network types, an airborne network requires a network controller to monitor queue sizes, network congestion, and flow priority and respond by making informed routing decisions for the routing and rerouting of information throughout a network. When the controller detects network congestion at any point within the network, the controller must adjust the routing throughout the network to reduce or eliminate network congestion. This section describes the design of the DNUC which controls network flow to optimize a network according to the updates the DNUC receives from other network devices.

The main goal of the DNUC is to show that Kalman filter predictions can be used to optimize network throughput in a dynamic environment where link speeds are constantly changing. The network controller can utilize these Kalman filter predictions to make informed decisions about the network. The network controller can optimize network flows to increase throughput. Even though an airborne environment is highly dynamic, a centralized device is necessary as to have awareness of the overall state of the network to make the necessary routing decisions. The centralized server could be a specific device or an elected leader that changes from time to time. For the case of this research, the ground station is the centralized server since every node communicates with the ground station already. Therefore, all network nodes have communication with the ground station. The main benefit of a centralized system is the ease of maintaining accuracy of network routing information and it allows other network devices to access a single location for all necessary network topology updates. Even though optimization has not taken place on the controller used for this research, the controller demonstrates validity of Kalman filter predictions in an airborne environment.

A key feature of the DNUC is the ability to receive timely updates from network devices noticing congestion. The controller can use these updates to generate informed network routing decisions. When a node predicts that the router queue reaches a predetermined threshold, the node updates the controller and requests new routing information from the controller. The controller then passes along the new route information to the requesting node.

3.7.1 Features.

The DNUC needs to maintain certain requirements to maintain a level of network flow. These requirements specify what the DNUC needs to do to optimize network routes. Chapter IV highlights the tests performed and demonstrates the functionality of the controller within the network described above. The four main features of the controller and their descriptions are as listed below.

- ***Network Congestion Detection*** is the ability of the controller to look at future queue size predictions made by the Kalman filter. Each node has an individual Kalman filter that allows each device to make predictions independently of another. The controller does not have direct access to the Kalman filter predictions, it is the responsibility of each node to use the Kalman filter predictions to decide when to update the controller and request route changes. All Kalman filters for this research were written using Matlab.

- ***Network Update Collection*** is the ability of the controller to receive updates from each node thus allowing the controller to make informed network routing decisions. When a node notices network congestion, an update request is sent to the DNUC. The controller requests the node to send the update. Once received, the controller informs the network route generation portion. The update function consists or two parts. First, the node itself informs the controller of the impending update. Second is the controller itself which responds to update requests and receive the updates. The update portion of the controller is a Python socket program.

- ***Network Route Generation*** is the controller's ability to parse the received updates from network nodes and determine the best possible route based on the latest update received from each node. The network generation looks through all updates to get awareness of the entire network state and determines the optimum route for each node. The network route generation portion is written in Python.

- ***Network Route Distribution*** is the ability of the controller to pass new routing information to all nodes when a node requests a new route. When the Kalman filter on a node notices that the amount of queued packets predicted is consistent with network congestion, the node requests a new route from the controller. The controller then passes along the updated route information back to the node. The network route distribution portion is a Python program.

All portions of the controller rely on the ability of the node to make queue size predictions. Even though the controller is constantly running, the controller waits for nodes to make routing updates and request new routing information. The Kalman filter Matlab program calls the update and route request Python scripts.

3.7.2 Design.

The DNUC receives network condition updates from every node in the network. The controller makes decisions based on these updates to optimize network traffic flow. When a node updates the controller, the controller potentially produces new routing information. An illustration of this process is shown in Figure 3.16. The network nodes receive queue size updates, passes the update to the Kalman filter that predicts the upcoming queue size. Prior to implementation, a decision about the two queue size thresholds is made. The two thresholds correspond to the predicted router queue size. When the predicted queue size exceeds threshold one, the nodes updates the controller. When the queue size exceeds threshold two, the node requests

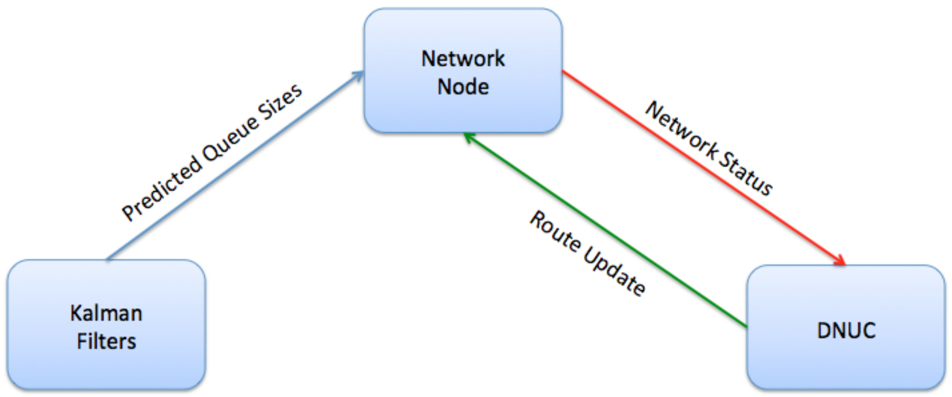


Figure 3.16. State Diagram of DNUC

updated routing information from the controller. For this research, threshold one is 100, threshold two is 200 and the maximum queue size is 500. These values allow the Kalman filter to accurately determine network congestion without burdening the network resources. For a specific node, if the predicted queue size remains below threshold one, no network congestion is noted and no updates are sent to the controller. With the nodes updating the controller only when congestion is noted, less non-data traffic is transmitted throughout the network. Every Node in the network relies on the Kalman filter process to make updates to the controller as well as receive updates from the controller. The network nodes follow the flow chart in Figure 3.17 to handle network congestion.

During normal operations, when the link can support all network traffic, a small number of packets may be in the network queue due to the transmission rates of the source. In the dynamic nature of airborne networks, congestion of network links can become an issue quickly. This issue means an accepted tolerance should be set low enough to ensure that if a link is degrading quickly, the controller change the routes quickly enough to allow normal data to continue to flow. For this research, network traffic is transmitted at a constant rate and the only reason the router queue should increase is when the link speed drops and can no longer support traffic load.

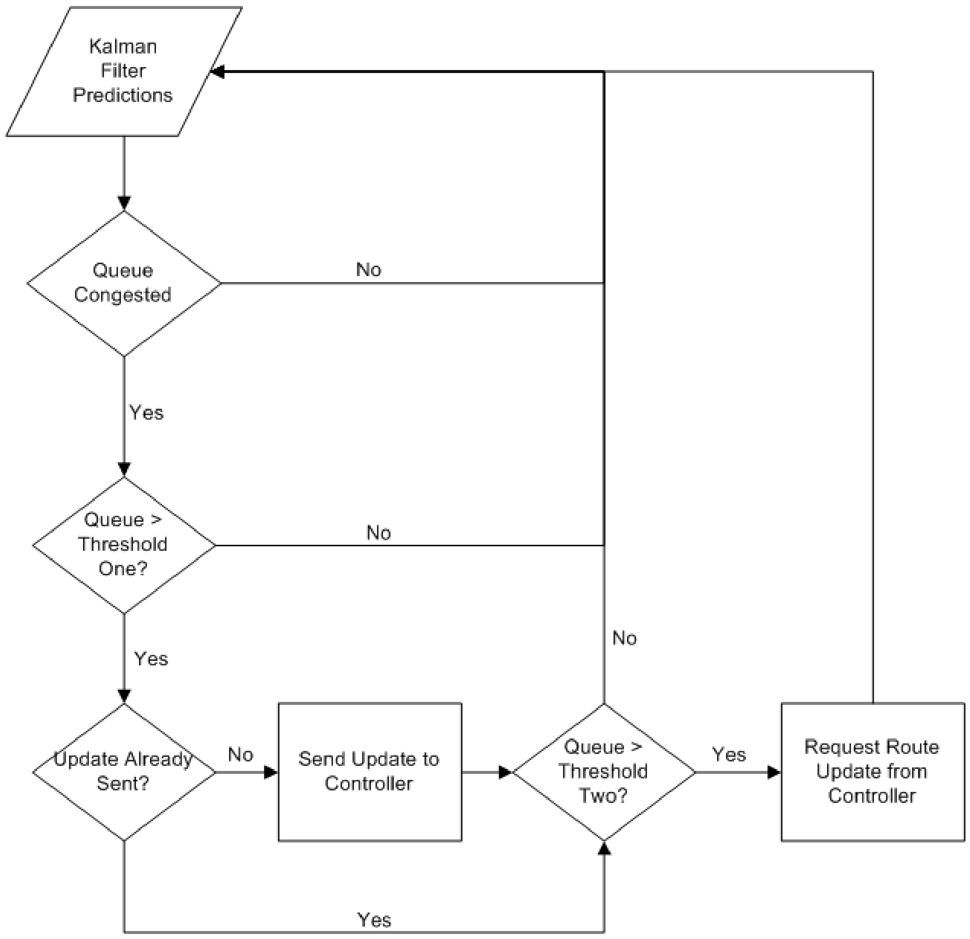


Figure 3.17. Flow Chart of Node Congestion Process

When the bandwidth of a link can no longer support network traffic, queued packets are seen in the router. The network node flow chart takes over from this point. When the node sends the update to the controller, the controller takes over to determine the best possible route at that moment. The controller reroutes traffic based on the following. The controller receives updates from network nodes. The algorithm parses the file looking at the first line to determine the IP address of the node sending the update. The file also contains the current gateway, all other network connections and their corresponding link speeds. The controller determines which route has the highest link speed and creates a file using the host's IP address and as the filename. The new file consists of one line and contains the new gateway and is formatted for installation directly onto the router. A new file is created every time a node makes an update to the controller unless a file exists for that specific node, as which time the file is overwritten with the new routing information. When the node requests updated route information, the controller determines which IP address is requesting the information. Along with the request, the node sends its IP address to the controller. The controller uses the IP address and sends the file with the same name to the node. When the node receives the new routing information, it loads the formatted file directly to the router, at which time, the router updates the routing table and traffic begins to flow through the new gateway.

3.8 Experiment

Along with DNUC, two different protocols are tested during this research: MME, and OSPF. MME was designed for use wireless networks where link states frequently change whereas OSPF was designed for use in networks with static link states. The MikroTik router came preinstalled with MME and OSPF. The number of dropped packets is captured when running DNUC. The number of dropped packets is captured while independently running MME and OSPF. For all three cases, no other dynamic routing protocols are active during testing.

This research consists of performing six different tests on each of the three different protocols with each test consisting of ten different runs. The six tests consist of two different UDP packet sizes with both of these tests run using three different delay schemes between packet transmissions: static delay time, Gaussian distribution of delay times and exponential distribution of delay times. The first packet size test consists of packets of average size of 992 bytes without the header. The two headers, UDP and IP, add an additional 28 bytes of data. The IP header consists of 20 bytes while the UDP header consists of 8 bytes. Including these headers, the total number of bytes in each data packets is 1020 bytes. With an average number of packets sent per second equal to 40, the 1020 bytes are an average data rate for the link speed configuration. The second packet size test consists of packets of size 1112 bytes without the header and 1140 bytes including the header. Again, 20 bytes for the IP header and 8 bytes for the UDP header. With an average number of packets sent per second equal to 50, the 1140 byte packets approach the limit of the network as configured. The data contained in the packets include an IP address, an asterisk and 980 or 1100 x characters (x = ASCII 78) respectively. The packets were put together in such a way so that a single packet could travel from the source to the destination without fragmentation. The Maximum Transmission Unit (MTU) for the network is set at 1500 bytes which means that fragmentation would occur on any packet larger than 1500 bytes which is why all packets are less than 1500 bytes. The two packet sizes are calculated to allow for two different delay times that are easily calculated while ensuring the packet sizes are below the MTU.

As listed above, packets transmission occurs at two different average rates. Transmission of the 1020 byte packets occurs at a rate of 40 packets per second resulting in an overall throughput at $40\frac{packets}{second} * 1020\ bytes * 8\frac{bits}{byte} = 326$ kbps. Transmission of the 1140 bytes packets occurs at a rate of 50 packets per second resulting in an overall throughput of $50\frac{packets}{second} * 1120\ bytes * 8\frac{bits}{byte} = 456$ kbps. The two different data rates test the ability of the different protocols to handle different traffic loads. The first test of 326 kbps tests the protocols at a average ($\approx 60\%$ of the maximum throughput) amount of data. The test that delivers 456 kbps tests the protocols at a high data rate which is $\approx 90\%$ of the total system throughput. As explained earlier, at any given time, each primary node can send 512 kbps to a secondary node. This means that the maximum throughput of any primary node to the ground station is 512 kbps. Therefore, the second set of tests, at 456 kbps, almost tests the maximum threshold of the system.

Three tests for each protocol are ran using the two different average rates with the three tests using different methods of sending data. Even though the transmission methods are different, the average transmission rate remains the same, 40 packets per second for 1020 byte packets and 50 packets per second for the 1120 byte packets. The first test has fixed delay times. For the 1020 byte packets, a packet is sent followed by a delay of 25 ms achieving a rate of 40 packets per second. For the 1140 byte packets, a packet is sent followed by a delay of 20 ms achieving a rate of 50 packets per second.

The second delay method, instead of the fixed delay, this test uses a random number generator following a Gaussian distribution. The 1020 size packet, the average of the Gaussian distribution is set at 0.025 making the average delay time 25 ms thus achieving the same number of packets per second on average. Figure 3.18 shows a Gaussian distribution with an average of 0.025 which is representative of what is seen when performing this test. The 1140 byte packets are similar. It uses the same random number generator except the average is set to 0.020. Figure 3.19 shows the same distribution of numbers except with an average of 0.020. If the random number generator chooses a negative number, the amount of delay is zero.

The third delay method uses a random number generator following an exponential distribution. For this method, the average delay between packets is again set to 25 ms and 20 ms for the 1020 byte packets and 1140 byte packets respectively. Figure 3.20 shows a random number generator following an exponential distribution with an average of 0.025. Figure 3.21 show what is expected from a random number generator following an exponential distribution with an average of 0.02. Again, if the random number generator chooses a negative number, the amount of delay is zero.

The three different delay methods described above are three different ways of

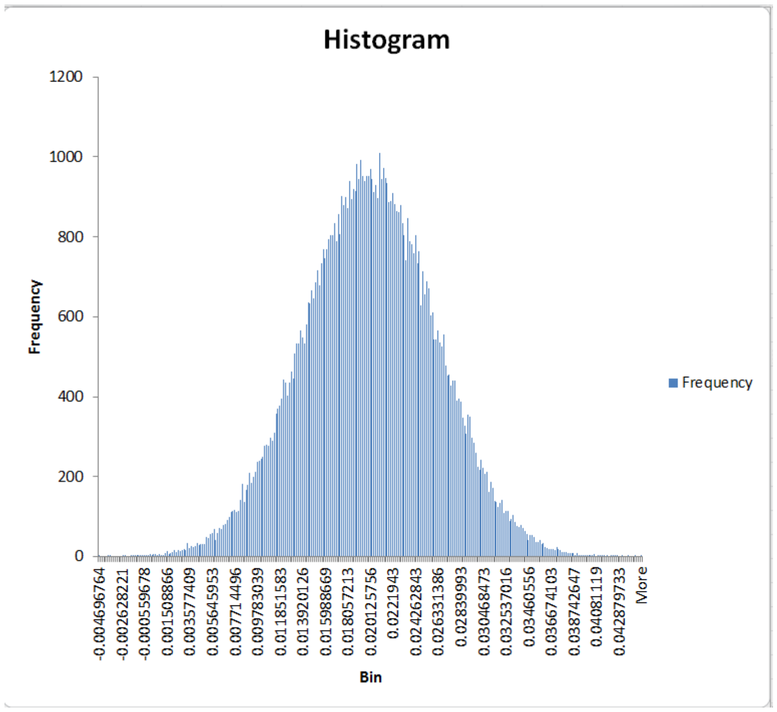


Figure 3.18. Histogram of Gaussian Distribution Wait Times for Gaussian 1

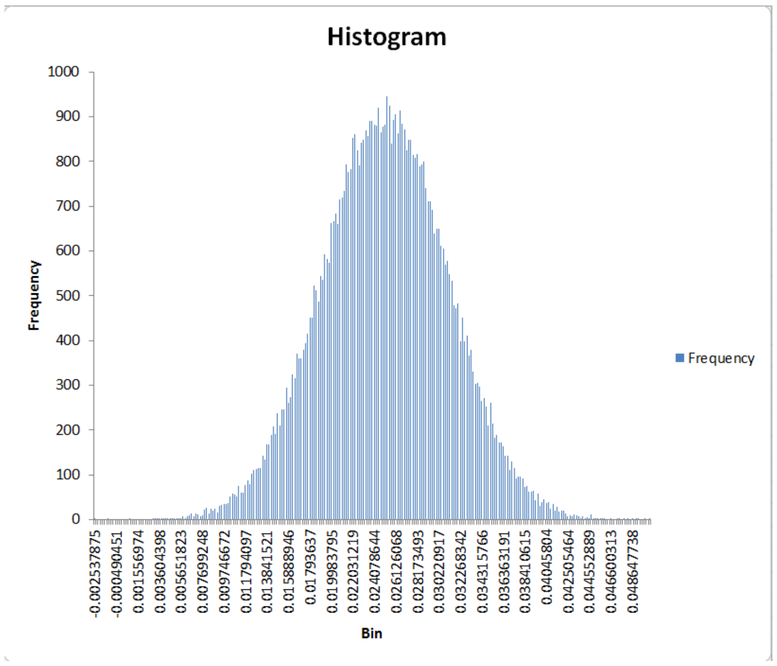


Figure 3.19. Histogram of Gaussian Distribution Wait Times for Gaussian 2

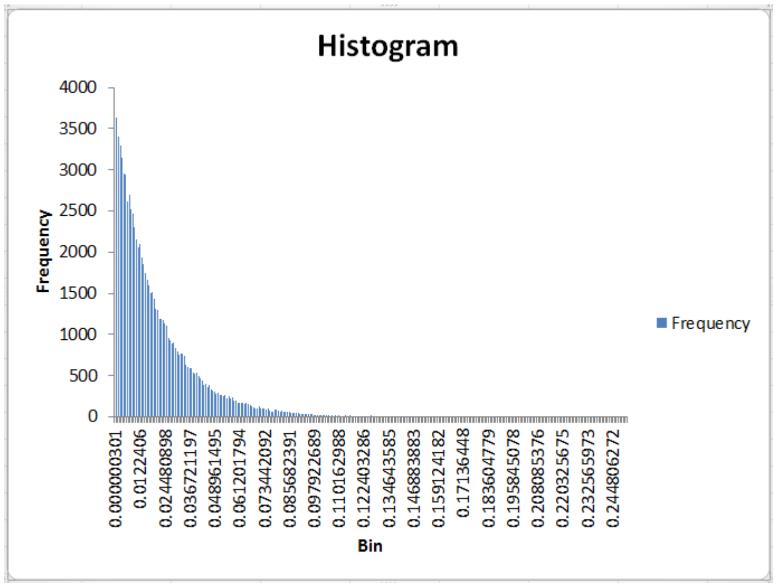


Figure 3.20. Histogram of Exponential Distribution Wait Times for Exponential 1

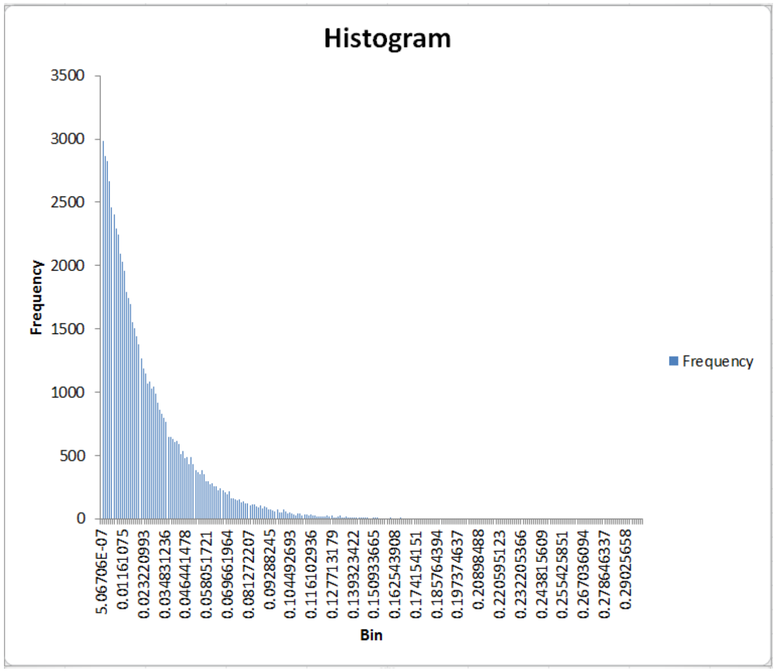


Figure 3.21. Histogram of Exponential Distribution Wait Times for Exponential 2

transmitting data. The static delay times are for comparison where the Gaussian and exponential distributions are more realistic of what is actually seen in networks. The Gaussian distribution delay times mimics real network traffic. When applications send network traffic, the amount of delay time between packet isn't an explicit amount of time. Some packets may take longer to package than others so the delay between packets is not constant. The exponential distribution of delay times is representative of bursty traffic. When bursts of traffic are sent, a large number of packets with small delays between packets, sent followed by a longer delay until more traffic is ready. All three methods deliver different results and provide more information for comparison.

To determine the number of packets lost during the tests, a python script is running on all nodes while a separate python script is running on the ground station. The node script tracks the number of times a packet is sent and annotates this number on a locally stored file. The source writes it's IP address inside the sent packets. When the packet arrives at the destination, the destination reads the data and IP address embedded in the packet to file stored on the destination. When all packets have been sent, the four primary node files and the ground station files are input into another script where a comparison is made between the number of times the primary node sent a packet and the number of times the ground station received a packet from each respective primary node. The total number of lost packets is the difference between the number of generated packets and the number of received packets.

Previous research has concentrated on the Kalman filter predictions [15]. This research concentrates on the network throughput for each setup. As described in Chapter 3, tests performed calculate the number of dropped packets for the duration of each test. Analysis of dropped packets concentrates on determining the average of dropped packets during each run. Tables and graphs showing the number of dropped packets are the main way of displaying the results for each test. The graphs display the number of dropped packets as well as a regression line for the data. Analysis of the data gathered from the three different protocols includes a comparison of the p-values from two-tailed t-tests.

The analysis of the data includes the calculation of the mean and standard deviation for each test. A graph of the 95% confidence interval to ensure that one protocol achieves better or worse results than another 95% of the time. Matlab completed all calculations and create all graphs.

3.9 Summary

In conclusion, the main concept of this chapter is to define how the airborne network is modeled and how various routing algorithms perform. The Kalman filter rating is a comparison of the predicted state and the observed state. The main concept of the DNUC is that how the controller receives updates from network nodes and determines the best possible route for traffic to travel from source to destination. The network nodes monitor the current and predicted queue sizes until the nodes discover network congestion. When the nodes notice congestion in the network and the queue size exceeds the first threshold, the node updates the DNUC. Once the queue size exceeds the second threshold, the node requests a new route. The DNUC is a centralized controller to track and update routing information. DNUC understands how to optimize the network to clear network congestion. Only when the DNUC receives information from network nodes can the controller make updated decisions. All decisions made by the network nodes and the controller itself are greedy and based on the current network state.

IV. Results and Analysis

4.1 Introduction

This chapter presents the results the tests performed on DNUC, MME and OSPF. This chapter consists of five different sections. The following list describes the contents of each section and describes the purpose of each test.

- Section 4.2 discusses the results of the experiment that demonstrates the functionality of the MikroTik router for use in limiting network link speed from the node to node within the network.
- Section 4.3 discusses the results demonstrating the performance of the UKF. These simulations establish a performance basis for the Kalman filters.
- Section 4.4 discusses the results using static delay times between packets. Each test compares the results from three different methods tested: DNUC, MME, and OSPF.
- Section 4.5 discusses the results using a Gaussian distribution for delay times between packets. Each test compares the results from the three different methods tested: DNUC, MME, and OSPF.
- Section 4.6 discusses the results using an exponential distribution for delay times between packets. Each test compares the results from the three different protocols tested: DNUC, MME, and OSPF.

4.2 Network Link Speed Validation

This research requires the use of variable network link speeds to prove Kalman filters work in dynamic networks. Without the actual hardware needed to test the

Kalman filter in networks with variable link speeds, this type of network requires emulation while using static networking devices. Variable link speeds can be simulated using the MikroTik router OS. Once configuration of the link speeds is complete and the MikroTik scheduler is configured, the link speeds change every minute. Validation of the link speeds needs to occur to ensure the network emulates an airborne network where link speeds are constantly changing.

Validation of the link speeds requires demonstration that the link speeds are constantly changing. Validation accomplishment takes place by using the ping command and ensuring the measured link speed is within 1% of the configured link speed. When the ping command sends a packet from a source to a destination, the RTT is measured and displayed. For validation of the link speeds and ease of calculation, a ping packet of size 524,064 bits is sent across a MikroTik router where configuration of different link speeds is complete. RTT Measurement of several different link speeds is shown in Table 4.2. The table lists the configured link speed, the RTT of the packet took to reach the destination and return to the source. Finally, calculation of the measured link speed was done by dividing the packet size by the RTT. For example $\frac{524{,}064}{4012ms} \approx 130,064bps$. As the results show, the configured link speeds are approximately equal to the measured link speeds. Since all measured link speeds are within 1% of configured link speeds, an acceptable range of tolerance is achieved and within an acceptable tolerance to show validity of this research.

Table 4.2. Network Link Speed Validation

Conf. Link Speed	RTT	Meas. Link Speed	% Difference
128 $kbps \approx$ 131,072 bps	4,012 ms	130,064 $bps \approx 127.02kbps$	99.2%
256 $kbps \approx$ 262,144 bps	2,005 ms	261,379 $bps \approx 255.25kbps$	99.7%
512 $kbps \approx$ 524,288 bps	1,002 ms	523,018 $bps \approx 510.76kbps$	99.8%
768 $kbps \approx$ 786,432 bps	667 ms	785,703 $bps \approx 767.29kbps$	99.9%
1 $Mbps \approx$ 1,048,576 bps	502 ms	1,043,563 $bps \approx 0.995Mbps$	99.5%

4.3 Kalman Filter Validation

The router queue size predictions from the UKF in each node controls the DNUC. Alqahtani's research further developed the idea of a controller that used UKF predictions [7]. In order to expand the previous research mentioned in Chapter II, all primary nodes contain UKFs. These nodes require Kalman filters due to the dynamic nature of the link between the primary and secondary nodes. When more data is transmitted across links that are unable to transmit the network traffic, the packets are placed in the queue of the router and router queue size increase. All outbound interfaces of the primary nodes have queues. The MikroTik router OS has predefined queues used during this research. Independent of packet size, the maximum capacity of the queues for this research is set to 500 packets. Each Kalman filter predicts one second into the future and makes a prediction every second. During the experiments, queue size values (actual queue size, UKF predicted value and naïve predicted value) were written to a locally stored file and used to verify the operation of the UKF. The naïve predictions are the previous value of the actual queue size. Comparisons were made between the UKF prediction, the naïve prediction and the actual queue size for the same time. These comparisons are used to compare the UKF predictions and naïve predictions and determine if the UKF is viable for use during this research. All Kalman filters were designed and configured the same, so testing one filter ensure that all filters are predicting as expected.

Table 4.3 displays the results of test 1 of the UKF. The results display the actual router queue size for each second, the UKF predicted and the naïve predicted queue size for the same second. The fourth column lists the difference between the actual router queue size and UKF predicted queue size along with the mean and standard deviation for the values. The fifth column displays the difference between the actual queue size and the naïve predicted queue size along with the mean and standard

Table 4.3. Queue Size - Actual vs Kalman Filter and Naïve Predictions

Actual	Kalman	Naïve	Actual - Kalman	Actual - Naïve
50	48	42	2	8
57	53	50	4	7
65	62	57	3	8
67	64	65	3	2
67	67	67	0	0
75	71	67	4	8
82	80	75	2	7
89	86	82	3	7
91	89	89	2	2
99	96	91	3	8
108	104	99	4	9
134	125	108	9	26
134	131	134	3	0
142	139	134	3	8
144	142	142	2	2
151	149	144	2	7
		Mean	3.0625	6.8125
		Std Dev	1.8786	6.0135

deviation for these values. The table shows that UKF predictions are typically more accurate than naïve predictions. The graph shows a couple instances where the naïve prediction is more accurate than the UKF.

Table 4.4 displays the results of test 2 of the UKF. The results display the actual router queue size for each second, the UKF predicted and the naïve predicted queue size for the same second. The fourth column lists the difference between the actual router queue size and UKF predicted queue size along with the mean and standard deviation for the values. The fifth column displays the difference between the actual queue size and the naïve predicted queue size along with the mean and standard deviation for these values. The table shows that UKF predictions are typically more accurate than naïve predictions. The graph shows a couple instances where the naïve prediction is more accurate than the UKF.

Table 4.4. Queue Size - Actual vs Kalman Filter and Naïve Predictions

Actual	Kalman	Naïve	Actual - Kalman	Actual - Naïve
53	48	45	5	8
59	55	53	4	6
66	62	59	4	7
76	72	66	4	10
83	78	76	5	7
83	82	83	1	0
90	88	83	2	7
100	95	90	5	10
107	105	100	2	7
155	141	107	14	48
163	157	155	6	8
172	166	163	6	9
183	177	172	6	11
187	181	183	6	4
194	189	187	5	7
203	199	194	4	9
212	207	203	5	9
		Mean	4.9412	9.8235
		Std Dev	2.7720	10.1627

According to Table 4.3 and Table 4.4, the mean for both tests show the UKF is predicting more accurate than the naïve filter. On test 1, when the actual amount of queue size is 134 and on test 2 when the actual amount of queue size is 155, the difference between actual and naïve is much higher during both instances. This is a result of the MikroTik router not creating the file used to read the number of packets in the queue for two seconds. This resulted in the same queue size value being read during two different seconds. If the file would have been written during both seconds, the queue size would have changed more gradual and therefore the difference between the actual and the naïve. The average of the UKF means is 4.00185 while the mean for the naïve filter is 8.318. These averages show that the UKF predictions are typically more accurate than the naïve predictions and therefore the controller makes routing decisions sooner.

4.4 Static Wait Time Tests

The first set of tests run use static delay times between packets. As stated in Chapter 3, each protocol, DNUC, MME and OSPF complete two Iterations. Iteration 1 has an average data rate while Iteration 2 has a high data rate. The data rates are considered average and high when compared to available throughput of the network used for these tests. Results are captured during each test and evaluated to compare the efficiency of the three different protocols.

4.4.1 Static Iteration 1 Results.

Static Iteration 1 consists of sending a 1020 byte packet followed by a delay of 25 ms. This makes the average number of packets sent every second equal to 40. The average data rate from a single node equal to 326 kbps. The network consists of four nodes that are sending data with each node have a data rate of 326 kbps. This makes the total throughput of the network equal to 1304 kbps. For static Iteration 1, the total number of packets sent from each node is 66,800 making the total number of packets sent equal to 267,200. Ten runs are performed on each protocol to ensure confidence in the results.

Table 4.5 displays the number of lost packets for each protocol for each of the ten runs of static Iteration 1. The table lists the number of dropped packets during each individual run along with the mean, standard deviation and percentage of dropped packets. Two-tailed t-tests are performed on all three results to determine the probability of null hypothesis, H, that the compared values have a statistical significance. The p-value when DNUC is compared to MME is 4.262e-15. The p-value when DNUC is compared to OSPF is 2.3464e-13. Finally, the p-value when MME is compared to OSPF is 6.6504e-08. Since all p-values are well below 5%, the null hypothesis is rejected for each comparison determining that there is no statistical

Table 4.5. Total Dropped Packets of Static Iteration 1

Run	DNUC	MME	OSPF
1	3,053	17,736	32,946
2	3,093	22,668	35,990
3	2,951	24,614	43,420
4	3,280	17,656	43,112
5	3,185	21,031	30,519
6	3,258	23,681	44,417
7	3,028	23,940	49,427
8	2,928	22,265	43,527
9	4,147	21,715	34,387
10	3,166	20,722	38,630
Mean	3,208.9	21,602.8	39,637.5
Std Dev	350.52	2,405.24	6,500.26
Dropped Packet Percentage	1.20%	8.08%	14.83%

significance between all data sets. The p-values corresponding to DNUC are much lower than the p-value comparing MME and OSPF.

Table 4.5 shows, DNUC dropped 85.15% less packets than MME and 91.90% less packets than OSPF. MME dropped 45.50% less packets than OSPF. Unlike MME and OSPF, DNUC uses the router queue size to determine network congestion, i.e., when link speeds are changing, and is able to react to the network congestion. Network congestion means that the link is unable to support the amount of network traffic being transmitted. As stated in Chapter 2, MME relies upon the sequence numbers of the packets. Without sequence numbers in the UDP packets, the amount of time MME takes to determine network congestion is much larger than that of DNUC. OSPF is constantly sending out updates to other routers in the network. This causes unnecessary network traffic to be transmitted across network links where bandwidth is already limited. OSPF is typically not found in dynamic networks and therefore in this type of network, OSPF is unable to change routes as frequently as MME or DNUC and performs inadequately when compared against these two protocols.

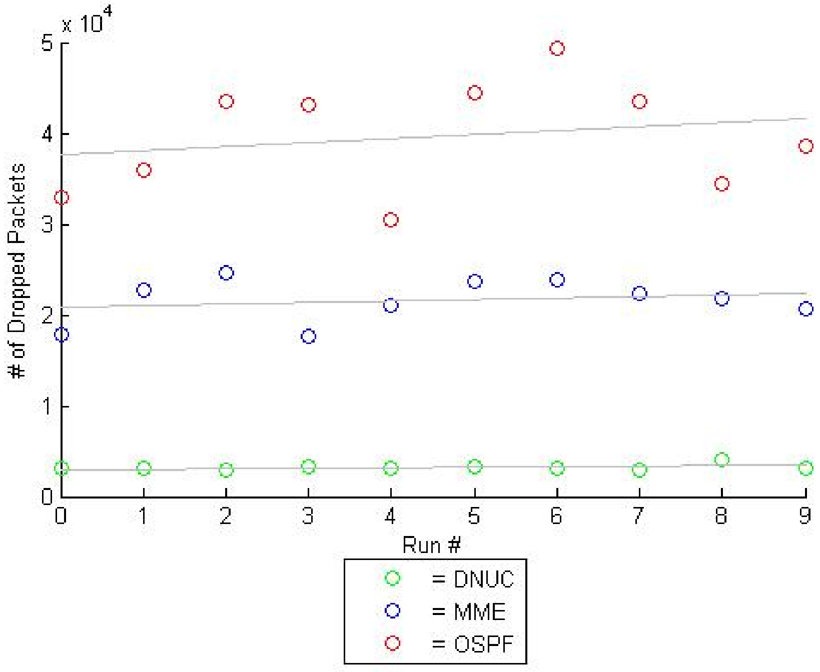


Figure 4.22. Scatter Plot of Static Iteration 1 Results

Figure 4.22 displays the number of dropped packets in a scatter plot along with a regression line to more easily compare the results and determine the future trend of packet loss. The regression lines show that all three protocols tend to have higher packet loss as time continues. Even though all trends are positive, OSPF and MME lose more packets much sooner than DNUC. Figure 4.22 supports the variability of the packets loss and standard deviation results shown in Table 4.5. The variability of packet loss of DNUC means that it consistently has low packet loss variability where the amount of packet loss demonstrated by OSPF is erratic.

4.4.2 Static Iteration 2 Results.

Static Iteration 2 consists of packets with a size of 1120 bytes and a delay time of 20 ms between each packet. This makes the average number of packets sent per second equal to 50. The average rate of data sent from a single node equals 448 kbps. Again, there are four nodes with a data rate of 448 kbps, this make the total throughput for the network equal to 1792 kbps. For static Iteration 2, the total number of packets sent from each node was 83,500 making the total number of packets sent across the network was 334,000. Again, ten runs where completed on each protocol to ensure confidence in the results.

Table 4.6 displays the number of lost packets for each protocol for each of the ten runs of static Iteration 1. The table lists the number of dropped packets during each individual run along with the mean, standard deviation and percentage of dropped packets. Two-tailed t-tests are performed on all three results to determine the probability of null hypothesis, H, that the compared values have a statistical significance. The p-value when DNUC is compared to MME is 1.8311e-18. The p-value when DNUC is compared to OSPF is 4.7468e-14. Finally, the p-value when MME is compared to OSPF is 3.0325e-05. Again, all p-values are well below 5% and therefore the null hypothesis is rejected for each comparison determining that there is no statistical significance between all data sets. The p-values corresponding to DNUC are much

Table 4.6. Total Dropped Packets of Static Iteration 2

Run	DNUC	MME	OSPF
1	2,450	58,083	55,146
2	2,270	52,757	95,291
3	2,342	49,384	75,608
4	2,318	55,279	76,534
5	2,744	50,471	71,283
6	2,622	60,556	71,613
7	2,325	51,008	66,831
8	2,473	51,513	83,209
9	2,395	60,330	76,128
10	2,713	60,052	87,548
Mean	2,465.2	54,943.3	75,919.1
Std Dev	171.12	4,466.28	11,147.1
Dropped Packet Percentage	0.74%	16.45%	22.73%

lower than the p-value comparing MME and OSPF.

The results again demonstrate that DNUC is more efficient changing network routes compared to MME and OSPF. When compared to the two existing protocols, DNUC dropped 95.51% less packets compared to MME and dropped 96.75% less packets compared to OSPF. MME dropped 27.63% less packets than OSPF. Again, these results are consistent with results of static Iteration 1. Static Iteration 2 sends 66,800 more packets, an increase of 25%, over static Iteration 1. With the increased number of packets and larger packet size, one would expect a larger number of dropped packets. MME drops an additional 33,340.5 packets average on static Iteration 2 when compared to static Iteration 1 while OSPF drops an additional 36,281.6 packets average on static Iteration 2 compared to static Iteration 1. DNUC on the other hand, drops 743.7 less packets average on static Iteration 2 when compared to static Iteration 1. The increased size and number of packets sent during static Iteration 2 had a different effect on the router queue size. The increased amount of traffic during static Iteration 2 allowed the queue to reach the saturation level more quickly and therefore, Kalman filter was able to determine the link degradation quicker during static Iteration 2. Therefore, DNUC was able to change the route more quickly. As stated in Chapter 3, the MikroTik router drops the packets currently queued when the link speed changes. If the Kalman filter can determine network congestion more quickly, the DNUC is able to change routes quicker than MME and OSPF. This allows more time for the link to transmit more of the queued packets before the link speed changes.

Figure 4.23 displays the number of dropped packets in a scatter plot along with a regression line to compare the results and determine the future trend of packet loss. The regression lines show that all three protocols tend to have higher packet loss as time continues. Even though the trends are all positive, OSPF and MME loses

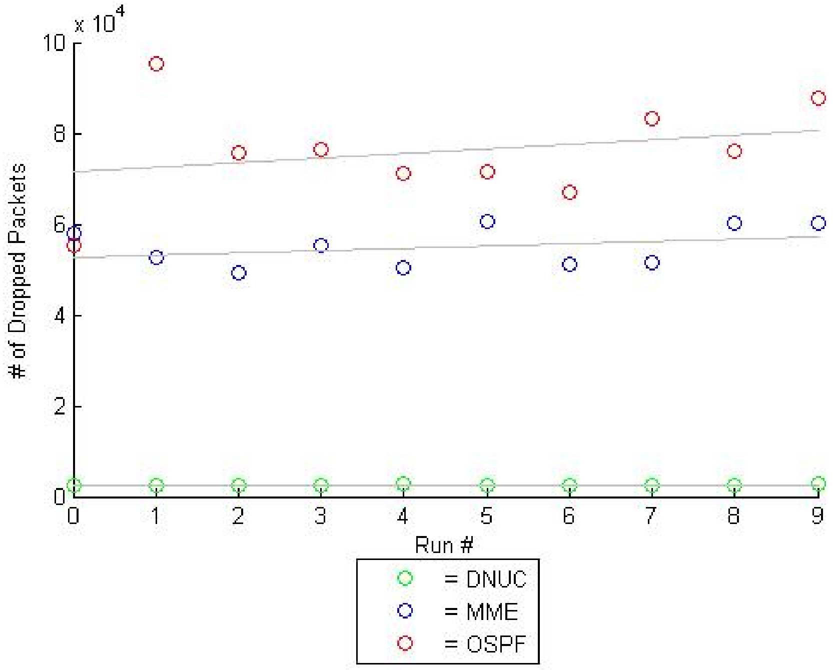


Figure 4.23. Scatter Plot of Static Iteration 2 Results

more packets quicker than DNUC. If the tests continued, MME and OSPF would tend to increase at a higher rate than DNUC. The variability of the packets loss is easily seen in Figure 4.23. The variability of packet loss of DNUC means that DNUC consistently has low packet loss where the amount of packet loss demonstrated by MME is highly dynamic and always higher. Even though the variability of MME is higher than that demonstrated by DNUC, the variability of MME is significantly lower than that of OSPF. Along with variability, overall packet loss is less with MME than that of OSPF.

Figure 4.24 shows the 95% confidence interval plot for both tests when using static interval times between packet transmission. The three data sets are color coded according to the legend. The blue line in each bar graph shows the 95% confidence interval of the mean for the two data sets while using static delay times. The longer the confidence interval is, the larger the standard deviation is for the data. The figure

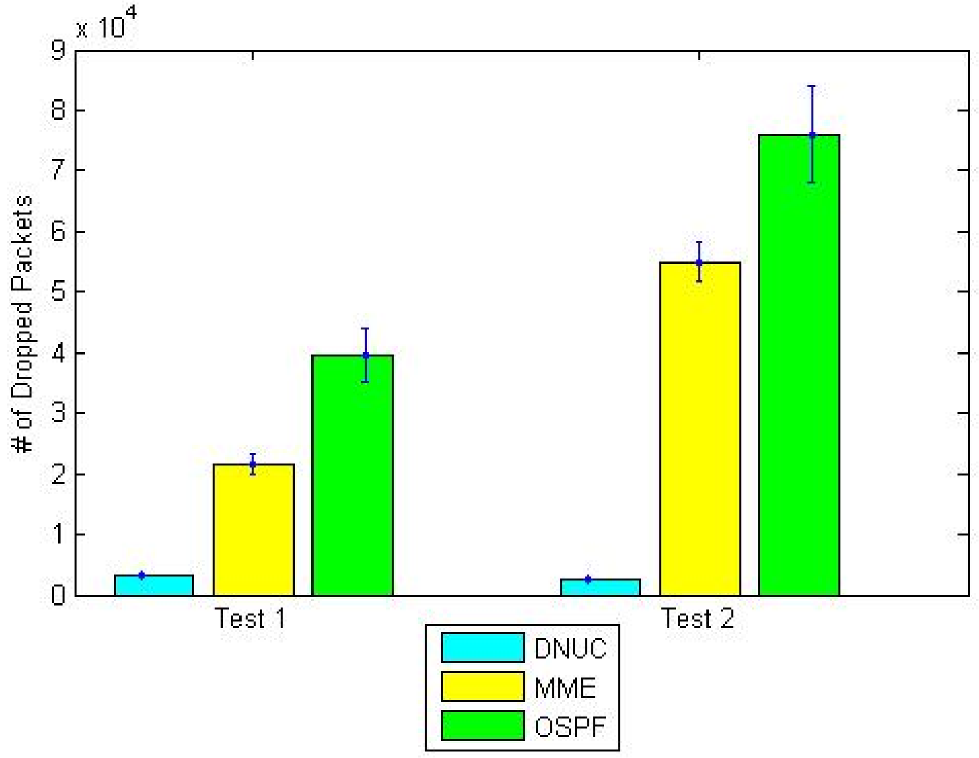


Figure 4.24. Confidence Interval of Static Delay Time Results

demonstrates that the confidence intervals between the three data sets do not overlap showing that on average DNUC outperforms MME and OSPF, and MME outperforms OSPF. The 95% confidence interval is in line with the standard deviations of the corresponding results which also demonstrate the variability of DNUC is much lower than that of MME and OSPF, and MME is lower than OSPF. Figure 4.24 validates the data listed above for static Iteration 1 and static Iteration 2.

Visual inspection of the tables and graphs for static wait times demonstrates that DNUC outperforms MME and OSPF. Between MME and OSPF, MME outperforms OSPF. Since MME works best in wireless ad-hoc networks while the MikroTik's version of OSPF works best in static networks, the results are accurate where MME outperforms OSPF in this network.

4.5 Gaussian Distribution Wait Time Tests

The second set of tests follow a Gaussian distribution for the delay times between packets. As with static delay time, each protocol, DNUC, MME and OSPF, are completing two Iterations of tests. Configuration of all elements of the tests aside from the delay times are the same as in Iteration one while using static delay times. Iteration 1 has an average data rate while Iteration 2 has a high data rate. The data rates are considered average and high when compared to available throughput of the network used for these tests. Results are captured during each test and evaluated to compare the efficiency of the three different protocols.

4.5.1 Gaussian Iteration 1 Results.

Iteration 1 consists of sending a 1020 byte packet followed by a delay of a random amount of time. The amount of delay time is determined using a Gaussian distribution (see Figure 3.18) with an average of 25 ms. This makes the average number of packets sent every second equal to 40. The average data rate from a single node is 326 kbps. The network consists of four nodes that are sending data with each node have an average data rate of 326 kbps. This makes the total throughput of the network averaged to 1304 kbps. For Gaussian Iteration 1, the total number of packets sent from each node is 66,800 making the total number of packets sent equal to 267,200. Ten runs are performed on each protocol to ensure confidence in the results.

Table 4.7 displays the number of dropped packets for each protocol for each of the ten runs of Gaussian Iteration 1. The table lists the number of dropped packets during each individual run along with the mean and standard deviation and percentage of dropped packets. Two-tailed t-tests are performed on all three results to determine the probability of null hypothesis, H, that the compared values have a statistical significance. The p-value when DNUC is compared to MME is 7.5175e-14.

Table 4.7. Total Dropped Packets of Gaussian Iteration 1

Run	DNUC	MME	OSPF
1	252	17,327	40,528
2	229	20,956	43,758
3	232	20,865	44,695
4	257	21,813	46,793
5	208	26,577	48,987
6	245	26,834	48,534
7	229	26,930	49,636
8	144	27,815	47,114
9	220	26,066	47,571
10	294	28,210	45,481
Mean	231	24,339.3	46,309.7
Std Dev	38.74	3,757.35	2,766.55
Dropped Packet Percentage	0.09%	9.11%	17.33%

The p-value when DNUC is compared to OSPF is 3.5840e-21. Finally, the p-value when MME is compared to OSPF is 1.4579e-11. Since all p-values are well below 5%, the null hypothesis is rejected for each comparison determining that there is no statistical significance between all data sets. Similar to static Iteration 1, the p-values corresponding to DNUC are much lower than the p-value comparing MME and OSPF.

As shown in Table 4.7, DNUC dropped 99.05% less packets than MME and dropped 99.50% less packets than OSPF. Finally, MME dropped 47.44% less packets than OSPF. The results again show that DNUC is performing much better than MME and OSPF and MME is performing better than OSPF. These results match up with that of static Iteration 1. One noticeable difference is the standard deviation of MME and OSPF. Unlike the static wait times, the standard deviation for MME is larger than OSPF. Even with an elevated standard deviation for MME, the standard deviation for OSPF is rather small. The small standard deviations mean that the number of dropped packets for MME is more widely scattered than those experienced by OSPF. DNUC experienced less packet loss for this test than any other test performed by several magnitudes over. As explained earlier, when the link speed changes, all packets in the queue are dropped. Using the Gaussian distribution for the delay times during Gaussian Iteration 1 allowed the queue to saturate quicker and therefore, the gateway changed faster allowing transmission for almost all packets in the queue of the previous gateway prior to the link speed changing.

Figure 4.25 show a scatter plot and regression line of the number of dropped packets for each protocol for Gaussian Iteration 1. As shown in the figure, the regression lines for MME and OSPF are again increasing quickly while DNUC is steady. For DNUC, the amount the regression line is increasing is negligible when compared to MME and OSPF.

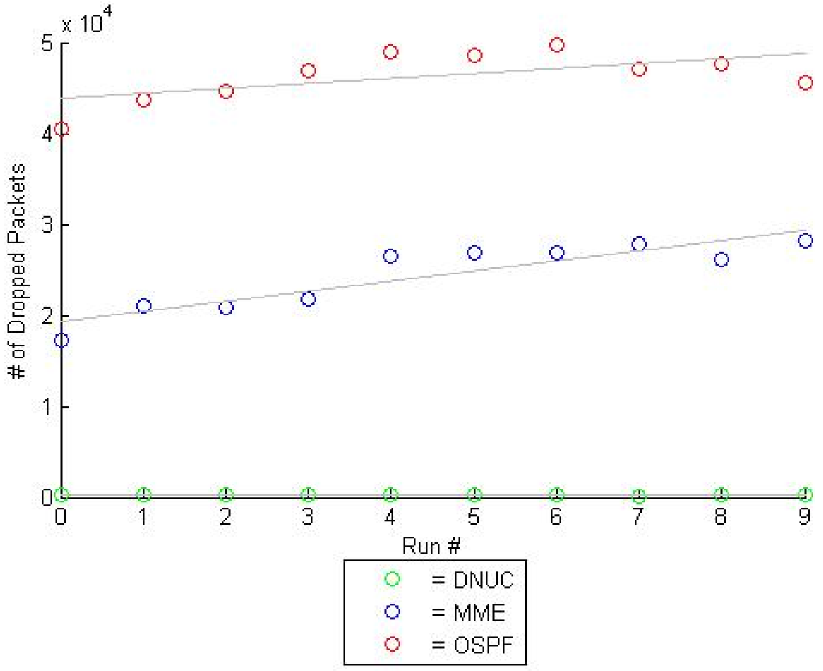


Figure 4.25. Scatter Plot of Gaussian Iteration 1 Results

4.5.2 Gaussian Iteration 2 Results.

Gaussian Iteration 2 consists of sending a 1120 byte packet followed by a delay of a random amount of time. The amount of delay time is determined using a Gaussian distribution (see Figure 3.19) with an average of 20 ms. This makes the overall average number of packets sent per second equal to 50. The average data rate from a single node is 448 kbps . The network consists of four nodes that are sending data with each node have an average data rate of 448 kbps. Making the total throughput of the network averaged to 1792 kbps. For Iteration two, the total number of packets sent from each node was 83,500 making the total number of packets sent across the network was 334,000. Again, ten runs are performed on each protocol to ensure confidence in the results.

Table 4.8 displays the number of dropped packets for each protocol for each of the ten runs of Gaussian Iteration 2. The table lists the number of dropped packets during each individual run along with the mean, standard deviation, and the percentage of dropped packets. Two-tailed t-tests are performed on all three results to determine

Table 4.8. Total Dropped Packets of Gaussian Iteration 2

Run	DNUC	MME	OSPF
1	3,460	66,624	78,774
2	3,853	60,401	71,854
3	4,314	58,221	68,642
4	4,065	59,307	60,455
5	5,030	57,481	58,165
6	3,705	62,887	78,470
7	3,833	46,983	85,451
8	3,768	54,739	79,038
9	4,301	83,635	95,787
10	3,358	75,789	72,944
Mean	3,968.7	62,606.7	74,958.0
Std Dev	487.47	10,538.0	11,222.1
Dropped Packet Percentage	1.19%	18.74%	22.44%

the probability of null hypothesis, H, that the compared values have a statistical significance. The p-value when DNUC is compared to MME is 8.8349e-13. The p-value when DNUC is compared to OSPF is 9.7556e-14. Finally, the p-value when MME is compared to OSPF is 0.0206. Similar to Gaussian Iteration 1, all p-values are well below 5%, therefore the null hypothesis is rejected for each comparison determining that there is no statistical significance between all data sets. The p-values corresponding to DNUC are much lower than the p-value comparing MME and OSPF when comparing MME and OSPF, the p-value is much more when compared to other Iterations.

As shown in Table 4.8, DNUC dropped 93.66% less packets than MME and 94.71% less packets than OSPF. Finally, MME dropped 16.48% less packets than OSPF. The results again show that DNUC performs much better than MME and OSPF and MME is performing better than OSPF even though the amount of packet loss MME had compared to OSPF is much closer than previous tests. These results match the results seen of static Iteration 2. Gaussian Iteration 2 sends 66,800 more packets which is an increase of 25% over Gaussian Iteration one. With the increased number of packets sent and the larger packet size, a larger number of dropped packets would be expected during Gaussian Iteration 2. Along with static Iteration two, the number of dropped packets for MME and OSPF are higher than Gaussian Iteration 1. The cause of this is the transmission of more packets every second and the size of the packets. One difference between static and a Gaussian distribution for the delay time, the amount of dropped packets for DNUC increase by a factor of 17 times. With a Gaussian distribution and the transmission amount of traffic during Iteration 2 led to more packets being left in the queue when the link speed changed. An interesting discovery during Gaussian Iteration 2 is the amount of standard deviation noted by MME and OSPF. The increase of standard deviation means that the difference between the number of packets dropped during each Iteration is higher than noticed using static delay times.

Figure 4.26 shows a scatter plot and regression line of the number of dropped packets for each protocol for Gaussian Iteration 2. As shown in the figure, the regression lines for MME and OSPF are increasing. For DNUC, the slope of the regression line on the graph is difficult to tell but if the regression line is increasing, the increase is negligible when compared to MME and OSPF. One interesting factor about Gaussian Iteration 2 was that in a few run, OSPF outperformed MME. The explanation can be the amount of variability with MME and OSPF. MME ranged from approximately 46,000 dropped packets to 82,000 dropped packets and OSPF ranged from approximately 58,000 dropped packets to 98,000 dropped packets. Gaussian Iteration 2 is the first test with that amount of disparity between the lower and upper bounds of dropped packets. Even though these two protocols have highly dynamic variability,

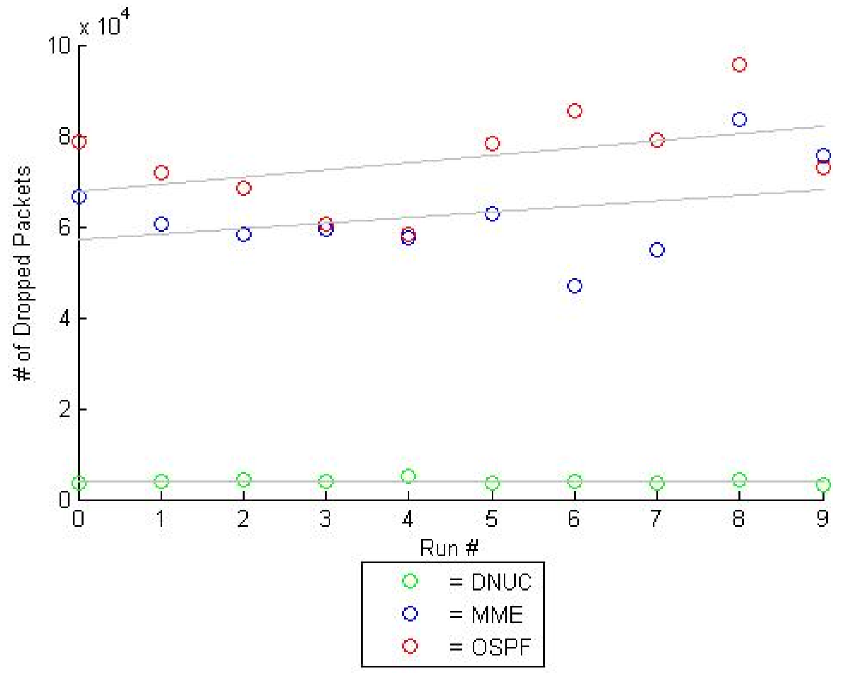


Figure 4.26. Scatter Plot of Gaussian Iteration 2 Results

DNUC is steady with very little variability.

Figure 4.27 shows the 95% confidence interval plot for both tests when using a Gaussian distribution for delay times between packet transmissions. The three data sets are color coded according to the legend. The blue line in each bar graph shows the 95% confidence interval for the corresponding data set. The figure demonstrates that the confidence intervals between DNUC, and MME and OSPF do not overlap while MME and OSPF partially overlap. The confidence interval graph also shows the high variability seen with MME and OSPF on Gaussian Iteration 2. For MME and OSPF, the 95% confidence interval is much larger on Gaussian Iteration 2 than Gaussian Iteration 1. This confirms the results seen above. The difference between Gaussian Iteration 1 and Gaussian Iteration 2 shows that DNUC performs a little

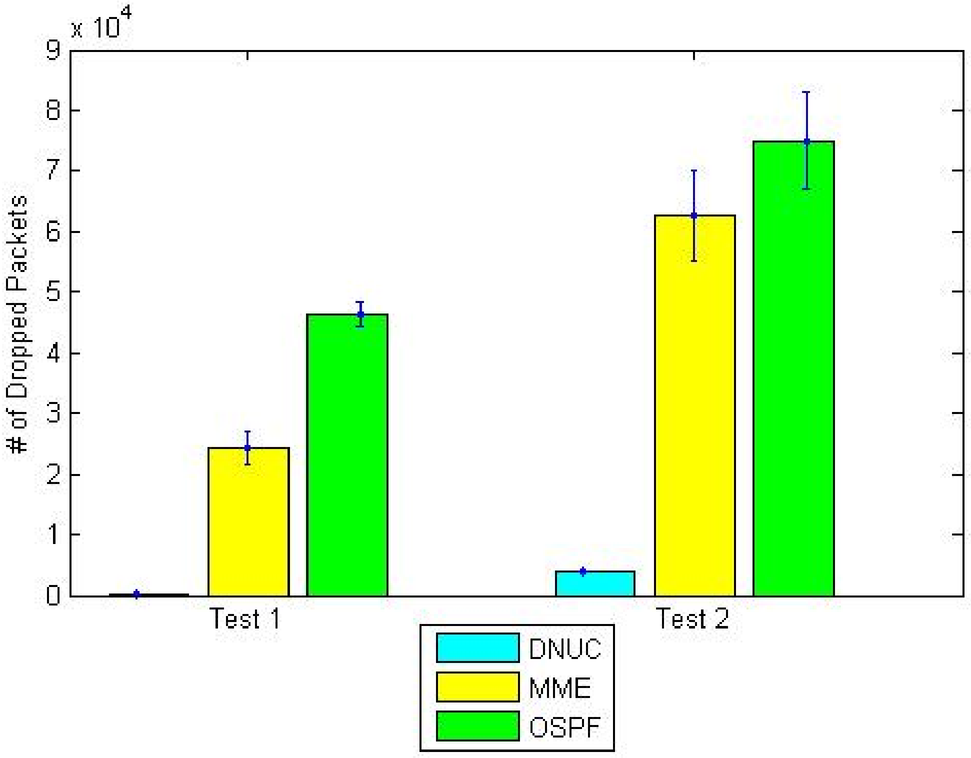


Figure 4.27. Confidence Interval of Gaussian Distribution Delay Time Results

better during Gaussian Iteration 1 compared to Gaussian Iteration 2 while MME and OSPF perform much worse during test two.

As in the static delay time tests, visual inspection of the tables and graphs demonstrate that DNUC outperforms MME and OSPF. Between MME and OSPF, MME outperforms OSPF a majority of the time. Since the design of MME makes use of wireless ad-hoc networks and the MikroTik version of OSPF was designed for use in static networks, the results make sense that MME outperforms OSPF in this type of network.

4.6 Exponential Distribution Wait Time Tests

The third set of tests completed follow an exponential distribution for the delay times. As with the static and Gaussian distribution delay times, each protocol, DNUC, MME and OSPF, are completing two iterations of tests. This test evaluates the capability of the three protocols to handle bursty traffic. Results include the number of dropped packets for the duration of each test. The same three protocols are tested, DNUC, MME, and OSPF. Results are captured during each test and evaluated to compare the efficiency of the three different protocols.

4.6.1 Exponential Iteration 1 Results.

Exponential iteration 1 consists of sending a 1020 byte packet followed by a delay of a random amount of time. The amount of delay time is determined using a exponential distribution (see Figure 3.20) with an average of 25 ms. This makes the average number of packets sent every second equal to 40. The average data rate from a single node is 326 kbps. The network consists of four nodes that are sending data with each node have an average data rate of 326 kbps. This makes the total throughput of the network averaged to 1304 kbps. For exponential iteration 1, the total number

Table 4.9. **Total Dropped Packets of Exponential Iteration 1**

Run	**DNUC**	**MME**	**OSPF**
1	757	23,619	31,611
2	1,396	22,206	45,138
3	1,124	27,155	47,556
4	995	24,426	38,432
5	1,132	28,663	46,595
6	1,220	24,631	40,243
7	588	23,911	39,057
8	2,624	26,396	35,918
9	633	26,685	42,737
10	509	26,211	44,760
Mean	1,097.8	25,390.3	41,204.7
Std Dev	613.22	1,948.16	5,088.84
Dropped Packet Percentage	0.41%	9.5%	15.42%

of packets sent from each node is 66,800 making the total number of packets sent equal to 267,200. Ten runs are performed on each protocol to ensure confidence in the results.

Table 4.9 shows the number of packets lost during each run of iteration 1. The table lists the number of dropped packets during each individual run along with the mean, standard deviation, and the percentage of dropped packets. Two-tailed t-tests are performed on all three results to determine the probability of null hypothesis, H, that the compared values have a statistical significance. The p-value when DNUC is compared to MME is 1.4547e-18. The p-value when DNUC is compared to OSPF is 2.3751e-15. Finally, the p-value when MME is compared to OSPF is 3.2886-08. Since all p-values are well below 5%, the null hypothesis is rejected for each comparison determining that there is no statistical significance between all data sets. Similar to static iteration 1 and Gaussian iteration 1, the p-values corresponding to DNUC are much lower than the p-value comparing MME and OSPF.

As shown in the Table 4.9, DNUC dropped 95.68% less packets on average than

MME and 97.34% less packets on average than OSPF. MME dropped 61.62% less packets on average than OSPF. The mean for all three protocols is similar to what was seen in the results of the static and Gaussian distribution delay times tests. The results again show that DNUC is performing much better than MME and OSPF and MME is performing better than OSPF. These results match that of static iteration 1 and Gaussian iteration 1. The standard deviation in exponential iteration 1 shows that MME is more stable than OSPF. These results coincide with the static iteration 1 with one difference, MME had a standard deviation that was smaller than the other two tests. The percentage of dropped packets for exponential iteration 1 was lower than either of the previous two delay method tests. Since a majority of the delay times are smaller than the average, packets, on average, have a faster transmission rate than the previous two delay methods. The transmission rate means that when the link speed approaches the point when saturation, the number of packets being sent on average is large and the queue becomes congested quickly until a larger value is chosen for the delay and the queue size drops for that period.

Figure 4.28 show a scatter plot and regression line of the number of dropped packets for each protocol for exponential iteration 1. The regression line for MME and OSPF have a positive slope whereas DNUC is steady over the course of the ten tests. These results show similar trends as static iteration 2 and Gaussian iteration 2. Again, the variability of packet loss is seen in Figure 4.28 with the highest level of variability being OSPF where results range from 31,000 packets dropped to 48,000 dropped packets. These results are different than those seen from Gaussian iteration 1.

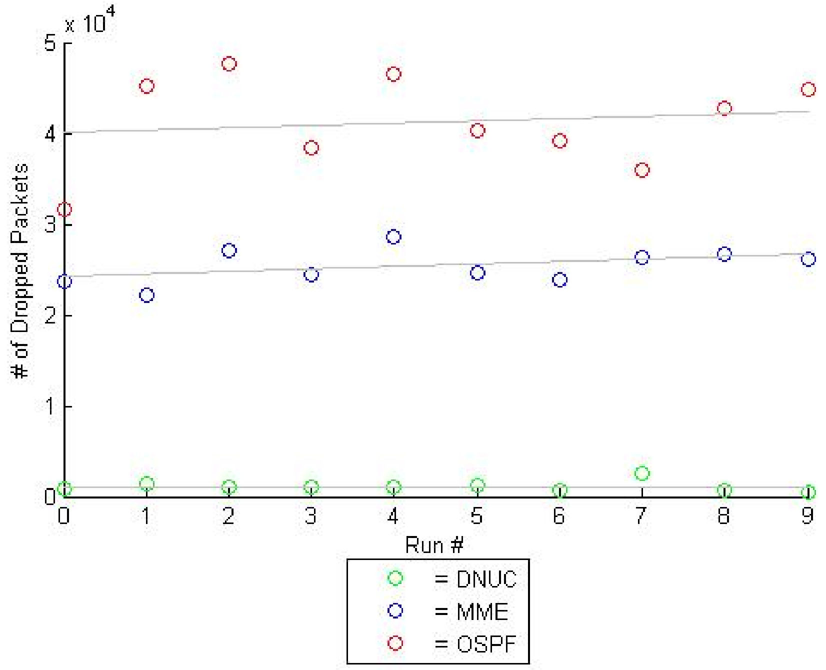


Figure 4.28. Scatter Plot of Exponential Iteration 1 Results

4.6.2 Exponential Iteration 2 Results.

Exponential iteration 2 consists of sending a 1120 byte packet followed by a delay of a random amount of time. The amount of delay time is determined using a Gaussian distribution (see Figure 3.19) with an average of 20 ms. This makes the overall average number of packets sent per second equal to 50. The average data rate from a single node is 448 kbps . The network consists of four nodes that are sending data with each node have an average data rate of 448 kbps. Making the total throughput of the network averaged to 1792 kbps. For iteration two, the total number of packets sent from each node was 83,500 making the total number of packets sent across the network was 334,000. Again, ten runs are performed on each protocol to ensure confidence in the results.

Table 4.10 displays the number of dropped packets for each protocol for each of the ten runs of exponential iteration 2. The table lists the number of dropped

Table 4.10. Total Dropped Packets of Exponential Iteration 2

Run	DNUC	MME	OSPF
1	2,891	63,189	70,727
2	2,169	52,281	71,418
3	2,703	57,027	106,547
4	1,650	63,353	98,372
5	2,485	51,190	75,937
6	2,820	57,279	84,696
7	2,701	59,144	98,762
8	2,895	66,680	91,304
9	2,044	66,309	73,612
10	2,692	60,570	90,110
Mean	2,505.0	59,702.2	86,148.5
Std Dev	417.63	5,369.16	12,846.3
Dropped Packet Percentage	0.75%	17.87%	25.79%

packets during each individual run along with the mean, standard deviation, and the percentage of dropped packets. Two-tailed t-tests are performed on all three results to determine the probability of null hypothesis, H, that the compared values have a statistical significance. The p-value when DNUC is compared to MME is 1.0886-17. The p-value when DNUC is compared to OSPF is 5.8816e-14. Finally, the p-value when MME is compared to OSPF is 1.1119e-05. Similar to exponential iteration 1, all p-values are well below 5%, therefore the null hypothesis is rejected for each comparison determining that there is no statistical significance between all data sets. The p-values corresponding to DNUC are much lower than the p-value comparing MME and OSPF when comparing MME and OSPF, the p-value is slightly higher when compared to other iterations.

The results demonstrate again that DNUC is more efficient changing network routes compared to MME and OSPF. DNUC dropped 95.80% less packets than MME and 97.09% less packets than OSPF. Finally, MME dropped 30.70% less packets than OSPF. Again, these results are consistent with results of all previous tests.

Exponential iteration 2 sends 66,800 more packets, which is an increase of 25% over exponential iteration 1. With the increased number of packets sent and the larger packet size, a larger number of dropped packets would be expected during exponential iteration 2. Similar to the tests when the delay times are static and use a Gaussian distribution, DNUC had a lower percentage of packets dropped during exponential iteration 2 when compared to exponential iteration 1. MME and OSPF had a larger number of dropped packets which is expected. As explained in the static delay time results section, with the increased amount of traffic and the large amount of small delay times, the router queue became saturated faster allowing the Kalman filter to predict network congestion faster. With the Kalman filter noticing the congestion faster, the DNUC changed the gateway faster and ultimately allow the transmission of more packets prior to the link speed changing and dropping the remaining packets in the queue. The amount of standard deviation is similar to those seen in previous results where the larger number of lost packets implies a larger standard deviation. With a larger average number of dropped packets, the results would display a larger disparity between the total number of dropped packets.

Figure 4.29 displays the number of dropped packets in a scatter plot along with a regression line to compare the results and determine the future trend of packet loss. The regression lines show that all three protocols tend to have higher packet loss as time continues. Even though the trends are all positive, OSPF and MME are dropping more packets at a quicker rate than DNUC. If the tests continued, MME and OSPF would tend to increase at a higher rate than DNUC. The variability of the packets loss corresponds to the the standard deviation results shown in the previous table. The regression lines show that for exponential iteration 2, OSPF was almost mirroring MME while the regression line of DNUC does not appear to rise. The figure shows that as in the static and Gaussian distribution delay time tests, OSPF has a

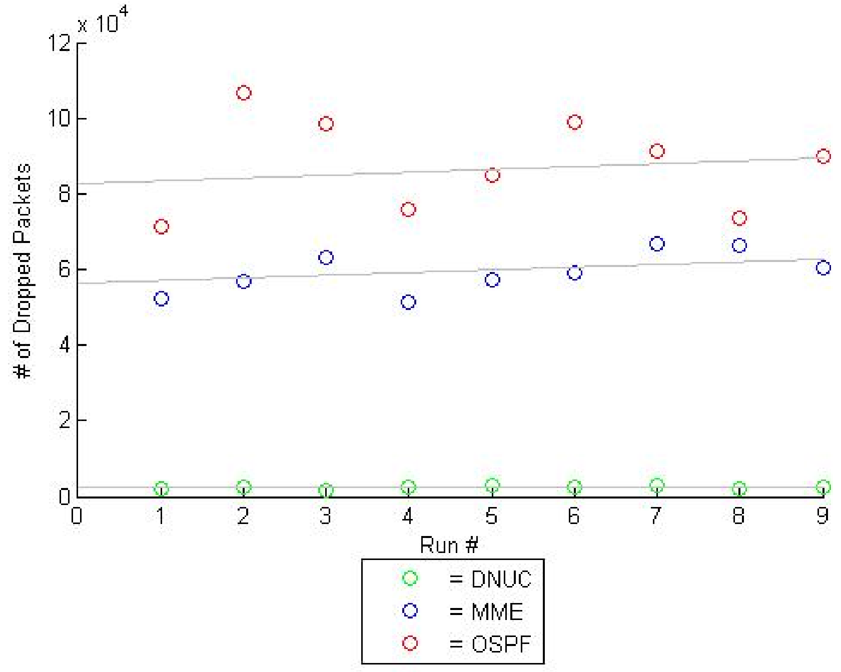


Figure 4.29. Scatter Plot of Exponential Iteration 2 Results

high amount of variability when compared to MME and DNUC.

Figure 4.30 shows the 95% confidence interval plot for both tests when using a exponential distribution for delay times between packet transmissions. The three data sets are color coded according to the legend. The blue line in each bar graph shows the 95% confidence interval for the corresponding data set. The longer the confidence interval is, the larger the standard deviation is for the data. The figure demonstrates that the confidence interval between the three data sets do not overlap. As shown in the 95% confidence interval plot, DNUC consistently has less packet loss than both MME and OSPF while MME has less packet loss than OSPF. The results match the results seen above. The difference between exponential iteration 1 and exponential iteration 2 shows that DNUC performs better during exponential iteration 1 compared to exponential iteration 2 while MME and OSPF perform much worse during exponential iteration 2. The standard deviations shown above have

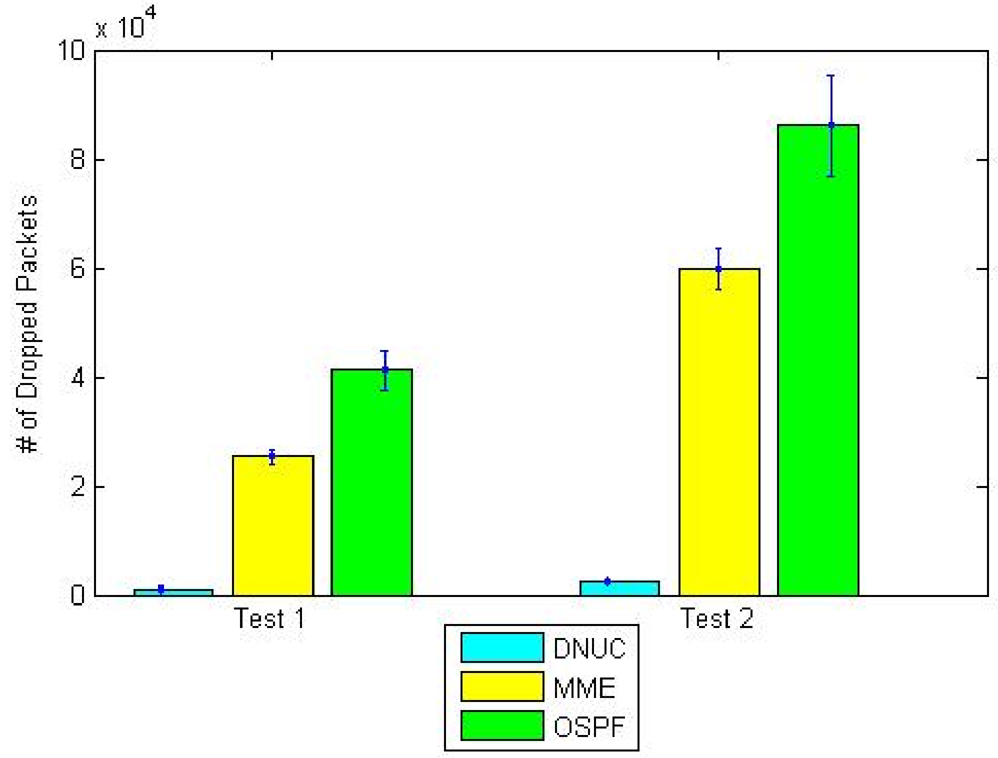


Figure 4.30. Confidence Interval of Exponential Distribution Delay Time Results

similar results with the confidence intervals for their corresponding data sets. The confidence interval matches the data above.

As in the static and Gaussian distribution delay time tests, visual inspection of the tables and graphs demonstrate that DNUC consistently has less packet loss than MME and OSPF in all cases. Between MME and OSPF, MME has less packet loss than OSPF a majority of the time. MME's design makes use of wireless ad-hoc networks and OSPF was designed for use in static networks, the results make sense that MME outperforms OSPF in this type of network.

4.7 Overall Results

The results obtained from the two different tests performed on three different protocols clearly show that DNUC performed better than MME or OSPF and, in most situations, MME performed better than OSPF. Since the MME design is for use in wireless networks and the MikroTik router OS version of OSPF is intended for use in static networks, MME should perform better in the network environment used for this research.

After thorough investigation, the majority of traffic loss for DNUC was due to the router dropping all queued packets when changing the link speed. Investigation also revealed that roughly half of the packet loss of MME and OSPF was due to this same reason. When performing tests on the three different protocols in this research, OSPF used one gateway to transmit data roughly 95% of the time and the network link that corresponds to the gateway OSPF is using is the highest capacity link roughly 33% of the time, OSPF does not use the optimal link 62+% of the time. The gateway that MME used to transmit traffic 90+% of the time was not the highest capacity link available. DNUC on the other hand, determined the highest capacity link and that interface was the gateway to transmit data until the Kalman filter discovered network congestion. When congestion was found, the controller changed the gateway to the interface corresponding to the highest capacity link found. Since DNUC is always using the gateway corresponding to the interface with the highest capacity to transmit data where MME and OSPF would rarely use the highest capacity link.

4.8 Conclusion

This chapter presents the analysis and results for the network traffic load tests and experiments run during the course of this research. Simulation results demonstrate that the overall network throughput increases with application of DNUC compared to

MME and OSPF routing protocols. Additionally, the test results indicate that overall network throughput increased with the application of the MME protocol compared to OSPF.

V. Conclusions

Military environments require highly dynamic MANETs to meet operational mission requirements. Military commanders rely on the timely delivery of critical battlespace information to make decisions quickly and accurately. Unfortunately, traditional MANET routing protocols do not provide the quality of service needed. Furthermore, they do not implement active controls to minimize the impact of network congestion. Using the router queue size along with the Kalman filter prediction concept, the possibility exists to optimize network routes to minimize over-utilization and increase network throughput.

5.1 Research Impact

Using the Kalman filter to predict queue size, this research demonstrated that the possibility to optimize performance of the highly dynamic networks exists. The DNUC implements network controls to provide quality of service in highly dynamic network environments. The increased information flow can assist in information superiority. The routing solution required the Kalman filter and DNUC to achieve success.

Even though the results demonstrate increased performance, this routing solution would not be appropriate for every scenario due to the limitations of the network components. Making considerations is necessary for the additional resources required to implement and achieve this solution. Certain computing capabilities are required for network nodes and the ground station. The nodes need to implement the Kalman filter and the ground station must run the DNUC to make routing decisions.

5.2 Contributions

As previously identified, the goals of this research were to show that airborne network environments can utilize the Kalman filter to discover degrading network links and develop an algorithm that can exploit these predictions to increase network throughput. The algorithm must satisfy the following criterion.

- Compute optimum routes based on predicted network traffic and topology.
- Use the network nodes and a Kalman filter placed on each node to monitor the network, and detect and react to queue size predictions thus minimizing network traffic loss due to link degradation.
- Minimize packet loss when compared to MME and OSPF routing protocols.

Demonstration of the algorithm's effectiveness was accomplished through performance of several tests running multiple simulations each. The tests show that DNUC performs better than MME and OSPF in all test cases. When the network traffic averaged 320 kbps, there were 93.6% less dropped packets compared to MME and 96.4% less dropped packets compared to OSPF. When network traffic averaged 450 kbps, there were 95.0% less dropped packets compared to MME and 96.2% less dropped packets compared to OSPF.

5.3 Future Work

Although results discovered from this research are encouraging, further refinement of methods specified in this document is encouraged. Specifically, future work should include the following.

- Additional scenarios: use real-world airborne radios for the network links and use actual distance between network nodes and determine if the results are similar.

- Compare the performance of this algorithm with other MANET protocols such as AODV, MANET extension of OSPF and OLSR.
- Develop an algorithm similar to multi-commodity flow to make use of the entire bandwidth available to every node.

5.4 Summary

This research demonstrates that the possibility exists to use the Kalman filter to predict network congestion and link degradation in dynamic network environments and make network route changes whenever the network link speeds from any node to all other nodes within communication range is known. Simulations demonstrate that routing solutions implementing Kalman filter techniques minimize packet loss and consequently increases network throughput.

Bibliography

1. Batman advanced documentation overview. World Wide Web Page. Available at *http://www.open-mesh.org/projects/batman-adv/wiki/Doc-overview*(accessed Dec 12, 2014).

2. Gig-battlefiled boundry. World Wide Web Page. Available at *https://me.stanford.edu/research/centers/ahpcrc/TA3Images.html*(accessed July 11, 2014).

3. Global information grid. World Wide Web Page. Available at *https://www.nsa.gov/ia/programs/global_information_grid*(accessed July 02, 2014).

4. Mikrotik manual. World Wide Web Page. Available at *http://wiki.microtik.com/wiki/Manual:TOC*(accessed Dec 05, 2014).

5. Network flows 3 - multicommodity flows. World Wide Web Page. Available at *http://www.cs.jhu.edu/˜scheideler/courses/600.348_F03/*(accessed Nov 11, 2014).

6. Performance of routing protocols for mobile ad-hoc networks. World Wide Web Page. Available at *http://w3.antd.nist.gov/wctg/manet/docs/perf_routing_protocols.pdf*(accessed July 22, 2014).

7. M. Alqahtani. Stochatic prediction and feedback control of routers queue size in virtual network environment. Master's thesis, Air Force Institute of Technology, 2014.

8. J. Betances. Context aware routing management architecture for airborne networks. Master's thesis, Air Force Institute of Technology, 2012.

9. I. Chakeres and L. Klein-Berndt. Aodvjr, aodv simplified. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(3):100–101, 2002.

10. B. Cheng and S. Moore. A comparison of manet routing protocols on airborne tactical networks. *Military Communications Confrence (MILCOM)*, 2012.

11. M. Compton. The Network Tasking Order (NTO). pages 1–7, San Diego, CA, 2008. IEEE.

12. S. Das, C. Perkins, and D. Belding-Royer. Ad hoc on-demand distance vector (aodv) routing. *Network Working Group*, 2003.

13. S. Dhurandher, M. Obaidat, and M. Gupta. A reactive optimized link state routing protocol for mobile ad hoc networks. *International Conference on Electronic Circuits and Systems*, 2010.

14. Soga T. Takenaka Y Terashima H. Mineno, K and T. Mizuno. Integrated protocol for optimized link state routing and localization: Olsr-l. *Simulation Modeling Practice and Theory*, 19:1711–1722, 2011.

15. J. Haught. Adaptive quality of service engine with dynamic queue control. Master's thesis, Air Force Institute of Technology, 2011.

16. S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear system. *Signal Processing, Sensor Fusion, and Target Recognition VI*, 3068, 1997.

17. L. Junhai, X. Danxia, and F. Mingyu. A survey of multicast routing protocols for mobile ad-hoc networks. *Communications Surveys and Tutorials*, 11(1):78–91, 2009.

18. R. E. Kalman. An introduction to the kalman filter. *Transaction of the ASME–Journal of Basic Engineering*, 3068:35–45, March 1960.

19. C. Kapoor and G. Sharma. To improve the qos in manets through analysis between reactive and proactive routing protocols. *Computer Science and Engineering: An International Journal (CSEIJ)*, 1(3):51–62, 2011.

20. M. Karimi and D. Pan. Challenges for Quality of Service (QoS) in Mobile Adhoc Networks (MANETs). In *Wireless and Microwave Technology Conference (WAMICON)*, Clearwater, FL, 2009. IEEE.

21. P. Maybeck. The Kalman filter: Introduction to concepts. *Stochastic models, estimation, and control*, 1979.

22. K. Mingook. Stochastic estimation and control of queues within a computer network. Master's thesis, Air Force Institute of Technology, 2009.

23. Network Working Group. *RFC5614 - Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding*, August 2009. Available at
http:tools.ietf.org/rfc/rfc5614.txt.

24. K. Pandey and A. Swaroop. A comprehensive performance analysis of proactive, reactive and hybrid manets routing protocols. *International Journal of Computer Science Issues*, 8(3), 2011.

25. J. M. Pecarina. Creating an agent based framework to maximize information utility. Master's thesis, Air Force Institute of Technology, 2008.

26. A. Penttinen. Chapter 8 - queuing systems. *Introduction to Teletraffic Theory*, 1999.

27. A. Sang and S. Li. A predictability analysis of network traffic. 39:329–345, 2002.

28. B. Shivahare, C. Wahi, and S. Shivhare. Comparison of proactive and reactive routing protocols in mobile adhoc network using routing protocol property. *International Journal of Emerging Technology and Advanced Engineering*, 2, 2012.

29. N. Stuckey. Stochastic estimation and contol of queues within a computer network. Master's thesis, Air Force Institute of Technology, 2007.

30. Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *ACM SIGCOMM Computer Communication Review*, 22:63–71, 1992.

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 26–03–2015	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2013 — Mar 2015

4. TITLE AND SUBTITLE Airborne Network Optimization with Dynamic Network Update	**5a. CONTRACT NUMBER** 14G260
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Paul, Bradly S., Capt, USAF	**5d. PROJECT NUMBER** 15G142
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-15-M-030
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research, Information and Life Sciences Directorate Attn: Robert J. Bonneau 875 N. Randolph St. Arlington, VA 22203-1768 (703) 696-6565(DSN: 426-62078) Robert.Bonneau@afosr.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR/RTA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Modern networks employ congestion and routing management algorithms that can perform routing when network routes become congested. However, these algorithms may not be suitable for modern military Mobile Ad-hoc Networks (MANETs), more specifically, airborne networks, where topologies are highly dynamic and strict Quality of Service (QoS) requirements are required for mission success. These highly dynamic networks require higher level network controllers that can adapt quickly to network changes with limited interruptions and require small amounts of network bandwidth to perform routing. This thesis advocates the use of Kalman filters to predict network congestion in airborne networks. Intelligent agents can make use of Kalman filter predictions to make informed decisions to manage communication in airborne networks. The network controller designed and implement in this thesis will take in the current and predicted queue size values to make intelligent network optimization decisions. These decisions will enhance the overall network throughput by reducing the number of dropped packets when compared with current static network and MANET protocols.

15. SUBJECT TERMS

Airborne network optimization,Dynamic Network Update Controller(DNUC),Mobile Ad-hoc Network(MANET),Queue size prediction

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj. Thomas. E. Dube, AFIT/ENG
U	U	U	UU	97	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636, x4613; thomas.dube@afit.edu