8-2018

Evaluation of IEEE 802.1 Time Sensitive Networking Performance for Microgrid and Smart Grid Power System Applications

Montie Edwin Smith Jr
University of Tennessee, msmit250@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Montie Edwin Smith Jr entitled "Evaluation of IEEE 802.1 Time Sensitive Networking Performance for Microgrid and Smart Grid Power System Applications." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Leon M. Tolbert, Major Professor

We have read this thesis and recommend its acceptance:

Qing Charles Cao, Fei Wang

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Evaluation of IEEE 802.1 Time Sensitive Networking Performance for Microgrid and Smart Grid Power System Applications

A Thesis Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Montie Edwin Smith Jr
August 2018

To my girlfriend:

Lizzie

To my family:

Mom and Wesley

In loving memory:

Edwin Smith

ACKNOWLEDGEMENTS

Thank you to Dr. Leon Tolbert and Dr. Fred Wang for your support and guidance during the course of my graduate studies. Thank you to Dr. Charles Cao for graciously agreeing to be on my defense committee. Thank you to CURENT and the EECS Department for the opportunity and resources needed to accomplish this project. Thank you to ARPA-E for funding in support of this work. Thank you to all fellow students and co-workers who provided valuable aid and insight through the course of my graduate studies. Thank you to Cisco for providing equipment used in support of this thesis. Thank you to National Instruments for providing guidance and assistance during the course of this project.

ABSTRACT

Proliferation of distributed energy resources and the importance of smart energy management has led to increased interest in microgrids. A microgrid is an area of the grid that can be disconnected and operated independently from the main grid when required and can generate some or all of its own energy needs with distributed energy resources and battery storage. This allows for the microgrid area to continue operating even when the main grid is unavailable. In addition, often a microgrid can utilize waste heat from energy generation to drive thermal loads, further improving energy utilization. This leads to increased reliability and overall efficiency in the microgrid area.

As microgrids (and by extension the smart grid) become more widespread, new methods of communication and control are required to aid in management of many different distributed entities. One such communication architecture that may prove useful is the set of IEEE 802.1 Time Sensitive Networking (TSN) standards. These standards specify improvements and new capabilities for LAN based communication networks that previously made them unsuitable for widespread deployment in a power system setting. These standards include specifications for low latency guarantees, clock synchronization, data frame redundancy, and centralized system administration. These capabilities were previously available on proprietary or application specific solutions. However, they will now be available as part of the Ethernet standard, enabling backwards compatibility with existing network architecture and support with future advances.

Two of the featured standards, IEEE 802.1AS (governing time-synchronization) and IEEE 802.1Qbv (governing time aware traffic shaping), will be tested and evaluated for their potential utility in power systems and microgrid applications. These tests will measure the latency achievable using TSN over a network at various levels of congestion and compare these results with UDP and TCP protocols. In addition, the ability to use synchronized clocks to generate waveforms for microgrid inverter synchronization will be explored.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction and Background

The world is growing more connected every day. Never before has there been such an abundance of information flowing over networks and around the planet, in addition to an abundance of new and different types of devices becoming connected. The so called "Internet of Things" (IoT) is becoming more prolific all the time. This does not only extend to the consumer space, but also to the commercial and industrial space.
One subset of the IoT specific to industry, the Industrial IoT (IIoT), has more stringent requirements than a Wi-Fi connected refrigerator. The industrial internet of things promises to lead to advancement in such industries as the aerospace, automotive, manufacturing, power, and more. Recent advancements in technology have not only made huge leaps in scale and bandwidth of industrial interconnection, but also to the fine degree of control and situational awareness possible [12]. These leaps in technology often require communication capabilities that previous open standards have been unable to provide. Current and future manufacturing processes and industrial automation applications require real time capabilities, deterministic latency, and synchronization of all nodes on the network to work at their full potential.

The ability for network connected devices to push the current boundaries of efficiency and cost reduction has led to the development of new IEEE standards to help with some new challenges. One such standard is the new IEEE standards that fall under the purview of Time Sensitive Networking (TSN). Previously, applications that required real time or deterministic capabilities could only be serviced by derivative or proprietary technologies, many of which are not compatible with each other or with more common communication standards. In addition, these solutions are often more expensive, and cannot scale or adapt as new technology improves. The new IEEE TSN standards are a new subset of standards in the 802.1 standard governing Local Area Networks (LAN), meaning that not only are the new capabilities open and nonproprietary, but they are 100% compatible with one of the most ubiquitous network communication standards in the world. This promises to not only make deploying IIoT networks with real time capabilities cheaper and easier, but to make these networks easier to integrate with existing infrastructure.

One area of interest where the benefits of TSN have yet to be fully explored is the power industry. New technology and communication developments have led to advances in how transmission and distribution lines are managed and controlled, leading to the concept of the *Smart Grid*. The transmission level of the grid has always been “smart” to some degree [13], but as communication technology and powerful computing and control has become cheaper and more widespread in industry, the idea of smart grid features trickling down into the distribution and residential levels has been explored. One area where these ideas culminate is the concept of a *microgrid*. A microgrid is a small network of loads and distributed energy resources (DER) that may either be connected to the larger power grid, act as an independent island, or somewhere in between. Microgrids have become an area of interest for research and industry with the recent trend in increased penetration of distributed renewable power sources (like residential solar installations) in the power grid. The increased levels of power being generated in smaller, more distributed locations instead of large localized power plants introduces new challenges in grid control and operation, but also new potential benefits. An area with a large number of distributed generation capabilities leveraged into a microgrid architecture need not worry that the transmission line to the nearest power plant was knocked out in a storm, as the microgrid can disconnect and continue uninterrupted. The ability of microgrids to intelligently route and manage their own power production is an intense area of interest in research and industry.

The necessary level of coordination, synchronization, and control required in a microgrid is an area where TSN may offer some benefits improving on current implementations. In an area of large distributed generation, the synchronization of inverters forming the microgrid must be highly precise to provide a stable grid. In addition, TSNs ability to prioritize traffic and provide real time control could allow residential inverter control to be sent and received over the same infrastructure as residential internet service. Furthermore, in a microgrid with little to no system inertia, the synchronization of distributed control actions throughout a microgrid becomes incredibly important. This is another area where the benefits of TSN's synchronization and latency guarantees may be beneficial.

1.1 MICROGRIDS

The current power grid is responding to technological advancements in our ever more connected world. Technology that offers the benefits of higher levels of safety, reliability, control, and cost effectiveness are being researched and implemented in the context of continuing the proliferation of smart grid technology. While the highest levels of the grid have always been to some extent smart in terms of automation and control, the continued fall in cost of power electronics and network connectivity has driven investment into smart technology for the feeder, distribution, and residential levels that have traditionally been thought of as a purely passive network [13].

These recent advances in technology have in part been driven by a surge in the adoption of renewable energy generation. Solar generation (PV) has seen the greatest amount of growth, as shown in Figure 1.1. In addition to the commercial and utility sectors, PV has exploded in the residential market in the past decade (see Figure 1.2), due in part to a more environmentally conscious populous and a sharp fall in cost of installation. Cost of solar installations has dropped more rapidly than anticipated year over year [1], becoming comparable to more traditional forms of energy generation in some markets with favorable government incentives or higher than average levels of solar irradiance. However, this increase in the proportion of generation supplied through the grid is not without its hurdles. High levels of distributed energy resources (DER) require new methods of control, protection, and integration with the larger grid network.

The microgrid concept is the point where the above two realms of research and development meet. The microgrid has the ability to leverage the penetration of renewables to improve overall reliability and system performance, while also giving grid operators more fine control at a smaller scale than otherwise would be available. The result is a grid with a more reliable and secure supply of energy, something that is becoming more and more important in our technological world [13]. In this introduction,

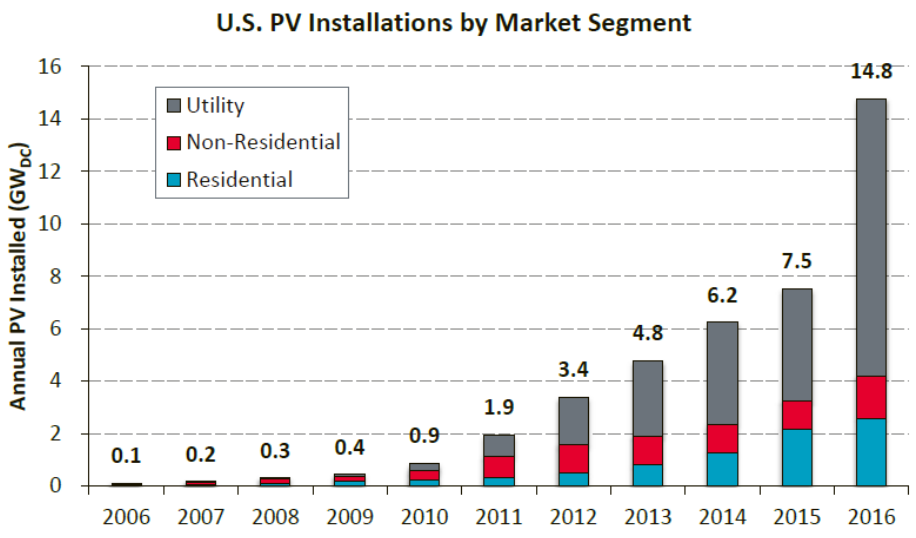


Figure 1.1: Growth in PV installation by market segment [1].

some of the current areas of microgrid research will be discussed, as well as the special considerations and benefits one must plan for when developing a microgrid.

1.1.1 What is a Microgrid?

A microgrid, as described in the book "The Microgrid Concept" can be thought of as an active power distribution network, rather than a passive network that is far more ubiquitous today [13]. Instead of a system that sends power directly from power plant to transmission line to feeder to residence, in one direction only, the microgrid allows for bidirectional flow based on need and circumstance. A microgrid, rather than being completely separate from the idea of a "smart grid", is in fact a type of smart grid.

The U.S. Department of Energy (DOE) defines a microgrid as follows: "A microgrid, a local energy network, offers integration of distributed energy resources (DER) with local elastic loads, which can operate in parallel with the grid or in an intentional island mode to provide a customized level of high reliability and resilience to grid disturbances. This advanced, integrated distribution system addresses the need for application in locations with electric supply and/or delivery constraints, in remote sites, and for protection of critical loads and economically sensitive development. (Myles, et. al 2011)" [14].

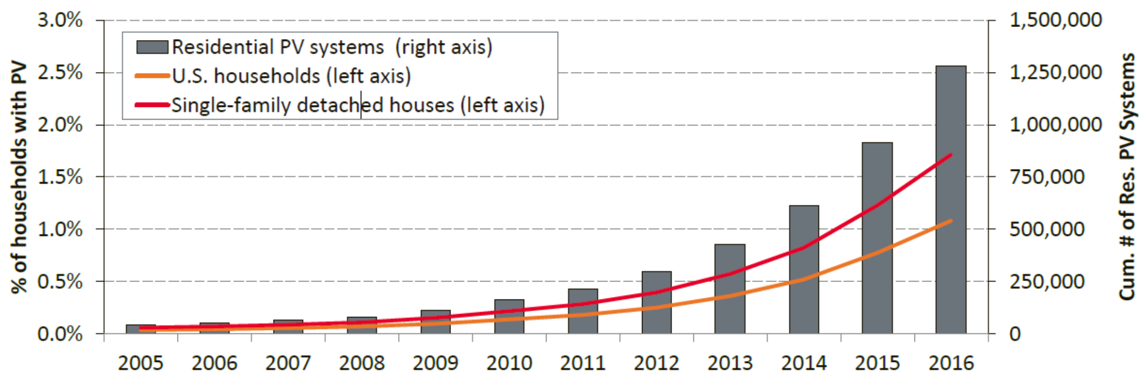


Figure 1.2: Growth in residential PV deployment [1].

The microgrid, like the smart grid, allows for high efficiency decision making, often automatically, to best serve the needs of both the operators and the consumers in real-time. Microgrids are all about new and revolutionary levels of grid control. The basic components of a microgrid include distributed energy resources (DER), large scale storage devices, flexible and controllable loads, and the control network necessary to operate all of these devices. In addition, some microgrids can provide an area's thermal needs, by using excess heat from the generation components to use for building heating or cooling (via absorption chillers) [13].

1.1.2 Benefits of a Microgrid

Microgrids offer numerous benefits to the service areas they power, as well as to the grid as a whole. Benefits for a microgrid can be split into two categories: economic/environmental benefits and reliability benefits [8]. The central feature of a microgrid that allows for its improved reliability and performance is its ability to act in both a grid connected or islanded mode. Grid connected mode is when the microgrid is connected to the main power grid via a single or multiple points of common coupling (PCC). Power can flow bidirectionally through this point, based on loads in the microgrid and its own levels of generation. Islanded mode is when the microgrid is disconnected from the main grid and supplying all its own power. The power during islanding mode can come from various DER installations, or from the battery energy storage systems (BESS) installed within the microgrid. This ability for the microgrid to operate independently from the main grid is what gives it its much higher levels of reliability. In the event of a natural disaster or other event which takes the grid down, the microgrid can operate for a period or indefinitely on its own [14].

The microgrid's ability to disconnect from the main grid also allows for other unique scenarios in addition to responding to a loss of main grid power. From the perspective of the grid operators, the microgrid can be treated as a large scale source of shed-able load or a dispatchable energy source. This gives grid operators greater flexibility in keeping the grid stable and in balance, as a microgrid can not only benefit itself but its adjacent grid area [14].

In addition, microgrids often incorporate renewable resources into their energy generation, reducing carbon dioxide emissions in their service area. Although the future microgrid concept generally assumes high penetration of renewables, other distributed energy sources such as diesel generators or gas turbines are also common in current microgrid implementations. Costs for electricity produced in the microgrid can be lower, due to peak-shaving/load-shifting ability enabled by having significant amounts of energy storage. Peak shaving allows a microgrid to store solar energy during high production but low demand (mid-day), and release this power back into the microgrid during the hours of low generation but high demand (evening). This also allows a microgrid to store energy from the grid during hours of low demand and low cost, and to become self-sufficient from its batteries during hours of high utility power cost. Both of these techniques allow a microgrid to reduce overall electricity costs to the system. On site generation in the microgrid also reduces the losses associated with long distance power transmission, as well as reduces costs due the fewer necessary distribution and transmission facilities [13].

Another benefit afforded by the microgrid is its ability to leverage excess heat from generation to be used for residential or commercial heating needs in the microgrid area (known as combined heat and power or CHP). By using excess heat which would have been otherwise released into the atmosphere for heating needs that would have been generated from electricity, overall effective system efficiency is increased. This is not possible with large scale power plants, due to their lack of proximity to the areas they often supply.

One other area which microgrids can provide significant improvements is faster response to changing loads by operators. In addition, there has been research done into ways to dynamically size the microgrid, to find the optimal balance between microgrid generation capacity and load. It should be noted that it is anticipated that most microgrids will generally operate in grid connected mode. Thus, it is expected that the benefits of a microgrid will be present not only in times of grid outage, but in everyday operation [13]. Peak shaving, excess heat usage, and greater reliability are all features that benefit the microgrid area in day to day operations.

1.1.3 Microgrid Control Topologies

There are three general control topologies for microgrid operation: centralized, decentralized and distributed control. Each has various drawbacks and advantages that must be considered.

A decentralized approach is the simplest, often having no higher level functions or communication links. In this type of microgrid topology, every inverter or distributed entity within the microgrid is under its own control. This type of topology is the least robust and dynamic. This topology usually operates with a droop-control mechanism for primary control. However, with no communication links, secondary voltage control and frequency regulation are generally poor [15, 16].

The distributed approach, unlike the decentralized topology, comes with some semblance of higher order coordination. This method often includes minimal communication links to facilitate coordination between the distributed resources, but no single entity is in control over the entire microgrid [17]. Every inverter or entity in this type of system takes its own corrective action based on information about the other entities in the microgrid [15]. This method is often chosen for its lower communication requirements and lack of a single controller as a single point of failure. However, with no single entity controlling all resources, more complex operating modes are not possible. This is often seen as a tradeoff to avoid a single centralized controller which could be seen as reliability concern.

A microgrid that incorporates centralized control has more global awareness of the entire microgrid, and thus is able to optimize for the entire microgrid for whatever goal the microgrid operators prioritize [8, 15, 18]. In addition, the addition of a single entity that controls the microgrid operation allows for complex modes not possible otherwise, such as blackstart and intentional islanding. However, this type of microgrid has higher communication requirements due to the need for a central controller. This has led to significant research into the decentralized approach, despite its inferior performance, due to concerns with the costs and reliability of a centralized microgrid requiring expensive high bandwidth communications [17]. Figure 1.3 shows the three types of microgrid topologies discussed.

Microgrid Control Hierarchy

Microgrid control is generally split into three levels: primary control, secondary control, and tertiary control [8, 15-17, 19]. Primary control governs the independent, immediate response by distributed generation entities to maintain synchronization and power sharing with a dynamic grid. It is the fastest loop, and does not require any communications. However, the primary control response is not perfect, and steady state errors cause the voltage and frequency to drift over time. This is where secondary control comes in. Secondary control (which can be communication assisted or independent) works to regulate voltage and frequency to their setpoint values relatively quickly (within seconds or minutes) to correct primary voltage errors. Tertiary control governs long term setpoints in a microgrid, and is primarily driven by economics.

1.2 TIME SENSITIVE NETWORKING

Time Sensitive Networking (TSN) is a suite of standards that are in the process of being added to the 802.1 Local Area Network (LAN) standards. They comprise a variety of standards that add support of improved time synchronization, deterministic latency control, redundancy and time-triggered actions. This thesis explores the IEEE TSN standards in the context of the 802.3 Ethernet standard, which is the physical layer technology on which the 802.1 LAN architecture operates.

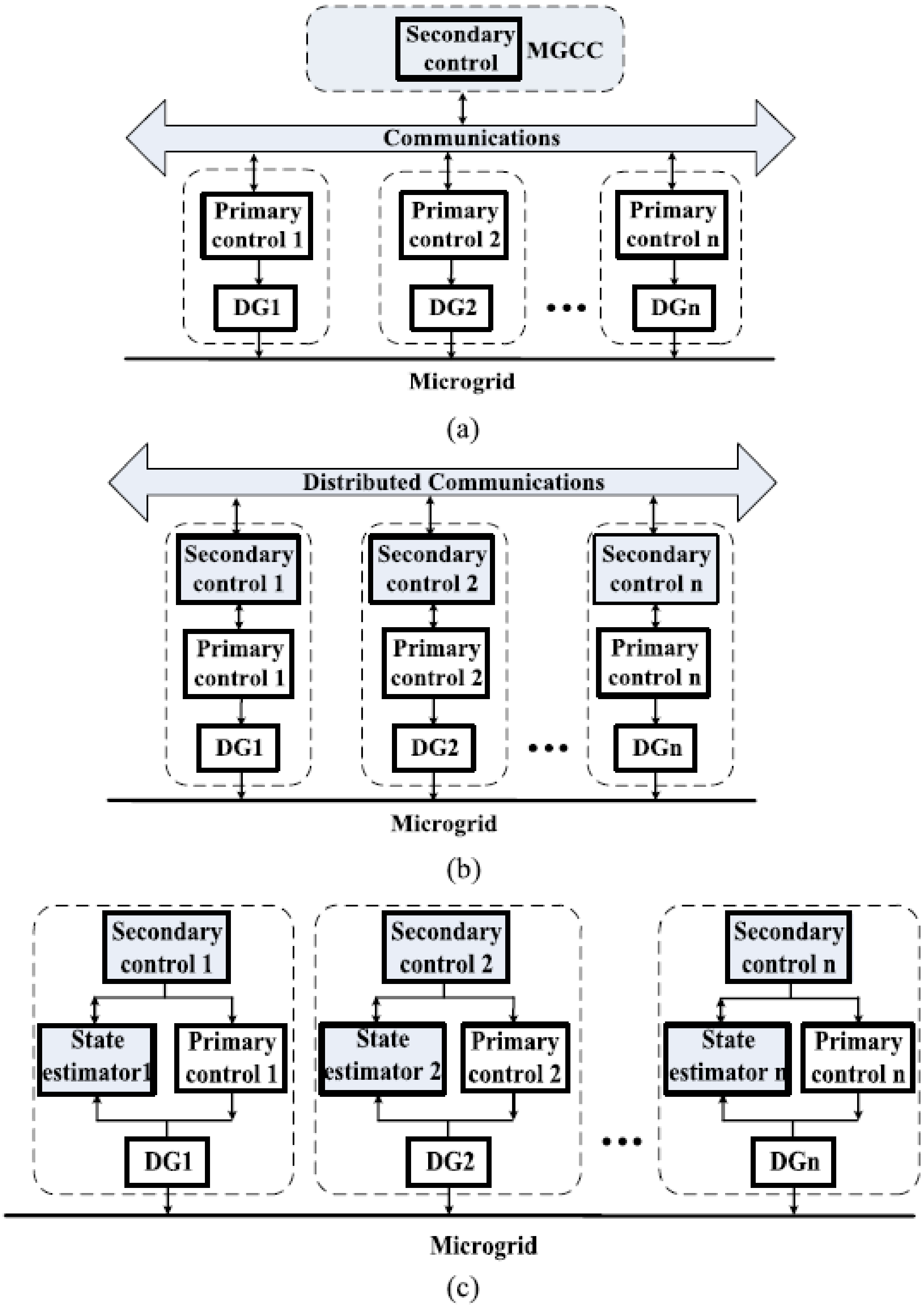


Figure 1.3: Centralized (a), Distributed (b), and Decentralized (c) microgrid topologies [9].

1.2.1 Audio Video Bridging – Precursor to TSN

Before introducing Time Sensitive Networking in detail, it is pertinent to introduce its direct predecessor: Audio Video Bridging (AVB). Research into AVB (as the name implies) began with an interest in using Ethernet networks for audio/video streaming applications that required tight latency requirements and synchronization [4, 12]. The task group, formed in 2005, focused more on Quality of Service (QoS) enhancements, rather than hard guarantees, as A/V streaming is often not a mission critical system.

AVB was created to allow "ad-hoc" plug and play compatibility between "talkers" and "listeners" through an Ethernet based network [2]. These talkers and listeners would request paths through the network with narrow jitter and latency requirements. As AVB was created with audio/video streaming in mind, the need for low jitter was greater than the need for low latency, and thus the mechanisms of AVB were to spread out traffic to help prevent interference from non-critical traffic. The main objectives of AVB were as follows [4]:

- Present a precise and common clock reference to the network
- Reduce latency and network delays
- Avoid interference from non-time critical network traffic

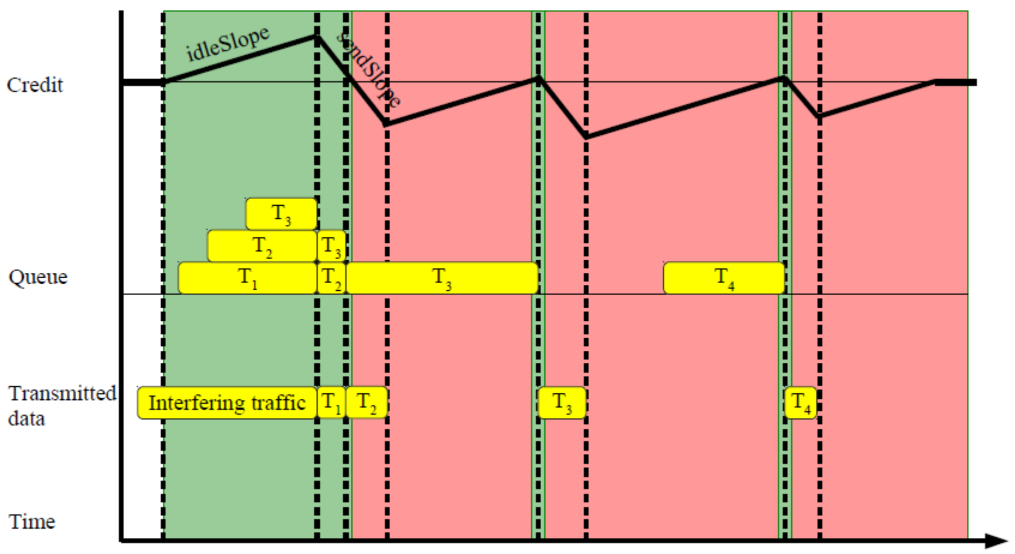


Figure 1.4: Graphical representation of AVB's credit based shaper [3].

The set of standards that fall under the original AVB task group include the following [2]:

- 802.1BA: Audio Video Bridging (AVB) Systems
- 802.1AS: Timing and Synchronization for Time-Sensitive Applications (gPTP)
- 802.1Qat: Stream Reservation Protocol (SRP)
- 802.1Qav: Forwarding and Queuing for Time-Sensitive Streams (FQTSS)

Together, the standards released by the AVB task group helped overcome some of the shortcomings present in an Ethernet based audio/visual network, such as synchronization of streams and delay due to congestion/buffering on the network [4].

Mechanisms of AVB Standards

The basic method by which an AVB system establishes a connection begins when an AVB listener requests a path through a network to an AVB talker. As part of the request, the listener would outline specific latency, jitter, and bandwidth requirements (using the stream reservation protocol (SRP) defined in 802.1Qat). This request is propagated from one switch to another along the path to the talker, while latency and jitter is calculated along each hop through the path [2]. The switches along the path utilize a *credit based shaper* (defined in 802.1Qav) to shape the traffic to meet the requirements outlined by the various requested streams through the switch. In AVB, unlike TSN, reducing jitter is generally more important than reducing latency (as in audio/video applications, a slight delay is more desirable than a jittery steam) [3]. The credit based shaper mechanism (shown in Figure 1.4) helps to spread out transmission of data, reducing bunches of data or bursts of data, as opposed to sending all data as soon as it is added to the transmission queue [2]. The common perception of time that all nodes on an AVB network share (utilizing the 802.1AS standard) allow the endpoints to conform to the established stream requirements [2]. Unlike TSN, where time-aware bridges administer traffic shaping, AVB relies on the endpoints themselves to adhere to the schedule defined by the credit based shaper administered by the switches.

AVB (as well as TSN) provides a major benefit not possible with other proprietary “real-time Ethernet” systems, in that it is built on top of 802.3 standard Ethernet, which is ubiquitous and nonproprietary. This also allows AVB/TSN to be backwards compatible (in terms of connection, but not features) to legacy 802.3 Ethernet systems. While AVB was originally designed with audio/visual applications in mind, some in industry saw that the features being developed could benefit other areas of industry as well. In 2012, after the culmination and release of their original mandated standards, the Audio Video Bridging task group was renamed the Time Sensitive Networking workgroup, reflecting their expanded scope away from A/V specific applications [2, 12]. With a new name, scope, and experience from developing the AVB standards, the workgroup set out with the goal to develop an expanded set of standards. The following changes were made with the goal to expand features and overcome drawbacks [2, 12]:

- Drop decentralized control in favor of centralized control/configuration
- Develop a new method of shaping to replace the inadequate credit based shaping method
- Expand feature set with a focus on the industrial segment, which has different requirements than the audio/video segment
- Reduce latencies and improve determinism accuracy
- Eliminate dependence on physical transmission rates
- The addition of fault tolerance without additional hardware
- Higher levels of security and safety
- Interoperability with other manufacturers' solutions

1.2.2 Applications of TSN

In addition to power system specific applications, there exist a plethora of other applications for the TSN Ethernet protocol suite to excel. The main industry that has seen significant research into TSN has been the automotive industry. As vehicles move toward autonomous operation, with more and more onboard systems needing to be interconnected, using Ethernet as a communication backbone has been a focus of intense research. In addition to critical systems related to autonomous driving, TSN Ethernet would allow for less critical systems, such as the navigation and infotainment systems, to be integrated to a common communication bus. The latency guarantees present in TSN allow for heightened levels of safety than would be otherwise possible with systems that could fight for bandwidth with the safety critical systems.

The aerospace industry has also taken an interest in TSN, for similar reasons as the automotive industry. On many commercial jets, each seat has an entertainment system that must be interconnected to watch movies or play games on long flights. TSN would allow these systems to connected over an Ethernet based network, while also allow for other more important systems (such as PA announcements) to take priority when necessary. In addition, this system would allow the audio/video portions of these streams to be synchronized throughout the cabin.

Many industrial applications have also been looked at as potential applications for TSN Ethernet. Robotic systems have been one system where the low latency and synchronized timing for data logging have been proposed as a use case that could benefit. In addition, any large distributed system that relies on data logging for measurement and control could benefit from receiving these measurements with greater levels of accuracy, coupled with a communication bus technology that is prolific and standardized.

1.2.3 Current Status

Currently, four of the seven TSN standards (IEEE 802.1Qbv, 802.1Qbu, 802.1Qca, and 802.1CB) have been published and are available. The remaining three (IEEE 802.1AS-Rev, 802.1Qcc, and 802.1Qci) are currently in draft stages. There exist several commercially available products that are enabled with the core functionality of the TSN suite (timing via 802.1AS (predecessor to 802.1AS-Rev) and latency determinism through 802.1Qbv). These include the National Instruments Compact RIO 9035-Sync and Cisco Industrial Switch 4000 switches that are utilized in the course of this thesis implementation. Though they do not support all TSN standards, the standards supported meet the minimum definition of a TSN enabled device [20]. Additional commercial products are being developed that plan to include the full suite of TSN standards from manufacturers such as Broadcom. In addition, there is a growing body of research available for both the AVB and TSN standards, from which novel methods of communication and control utilizing the standards have been published.

1.3 Project Description

In the experiments presented in this thesis, the latency guarantee features and synchronization precision of the TSN 802.1Qbv and 802.1AS standards are evaluated in the context of potential microgrid and power system applications. The performance of TSN latency guarantees are compared on a variably congested network with the common UDP and TCP protocols. The goal is to show that even under high levels of communication load, TSN latency guarantee features will allow the rapid transmission of data over a network, where non-TSN traffic would experience heavy delays or failures. In addition, the synchronization performance of the 802.1AS time-synchronization standard is explored as a potential means for generating synchronized signals to aid in the synchronization of distributed resources in a microgrid. In this experiment, the goal of demonstrating accurate synchronization within 1 degree of phase difference between two distributed units following a randomly time-varying unit will be tested.

1.4 Overview of Thesis Sections

In Chapter 1, an introduction to microgrids and the history of the Time Sensitive Networking standards have been presented. In Chapter 2, a detailed overview of the TSN standards used in this implementation are presented, including their mechanisms of operation. In addition, literature will be presented detailing areas where the benefits presented by the TSN standards may prove useful to various microgrid applications. In Chapter 3, the methods by which the experiments conducted are setup and executed will be outlined. In Chapter 4, the results of the experiments are detailed and compared. Chapter 5 provides conclusion to the preformed tests and recommendations for future research into TSN.

2 LITERATURE REVIEW

In the literature review to follow, the background and mechanics of the three TSN standards tested in this thesis are presented. Brief overviews are given of the remaining TSN standards that were unreleased or unavailable on commercial hardware at the inception of this project. Following the section on Time Sensitive Networking, the specific areas related to microgrids and smart grids that may benefit from the advantages TSN presents are reviewed.

2.1 CURRENT TSN STANDARDS

This section will focus on the three standards tested in the course of this project: IEEE 802.1AS, 802.1Qbv, and 802.1Qcc. The section regarding IEEE 802.1Qcc, which predominantly governs non-technical standards for the administration of TSN based Ethernet networks, is relatively brief. In addition, the remaining standards in the TSN suite (IEEE 802.1Qbu, 802.1Qca, 802.1CB, 802.1Qci) are briefly discussed, but they are not tested in the course of this project.

2.1.1 IEEE 802.1AS – Synchronization and Timing

The IEEE TSN standard regarding the synchronization of clocks in time-aware systems (TAS) is 802.1AS-Rev. Although 802.1AS-Rev is the successor to the 802.1AS standard released as part of AVB, they function much the same and rely on the same principles for realizing highly precise clock synchronization. IEEE 802.1AS-Rev adds several new features not present in 802.1AS, which will be detailed at the end of this section, after the common aspects of the clock synchronization process utilized by both standards is outlined. In addition, the Cisco Systems hardware used in this implementation are only 802.1AS compatible, and thus the focus of this description of TSN clock synchronization features is based on literature concerning 802.1AS. The standards (in the context of this thesis) are largely equivalent, in that both guarantee synchronized clocks to a high degree of precision for use by both the end user/devices, as well as in support of standards that make up the TSN suite.

The Precision Time Protocol (PTP), defined in IEEE 1588-2008 (also known as 1588v2), is a standard that aims to ensure clock synchronization on a network using physical layer timestamps [2, 11]. These timestamps are used to calculate network delay, and correct clocks throughout the system. IEEE 802.1AS (known as "Generalized Precision Time Protocol" or gPTP) is a profile of IEEE 1588, which reduces the overall number of options available compared with standard PTP, while increasing the extent of some physical layer options [2, 11]. As laid out in [2], the goal of the standard was to provide the following features for Audio Video Bridging applications:

- Provide performance specifications for switches as "Time Aware Bridges"
- Use accumulated "Neighbor Rate Ratio" calculations to improve accuracy and speed of convergence
- Include plug and play operation and startup with a specified Best Master Clock Algorithm (BMCA) used by switches. The BMCA is used to identify which clock on the network is most suited to be the grandmaster.
- Require the use of peer-to-peer transparent clocks and path delay processing
- Require two-step delay message processing (sync and follow-up messages)

As described in the standard, gPTP assumed that all time-aware communication between systems on the network would be conducted at the MAC layer (layer 2), while PTP supports methods for time-aware communication using Layers 2, 3, or 4 [21]. Additional differences between the standards can be found in Section 7 of IEEE 802.1AS [21]. Although IEEE 802.1AS is based on IEEE 1588, and is a profile of this standard, it can be viewed as a stand-alone document unto itself for the purposes of this overview [21].

IEEE 802.1AS Terminology and Topology

In an 802.1AS system, there exist two types of major components: end-stations and bridges (switches). Each of these components (also called time-aware systems) contain an internal clock, which will run fast or slow from the perspective of an ideal clock due to non-ideal oscillators, temperature variations, and other external factors [7, 22]. 802.1AS aims to periodically correct these clocks back to the reference time generated by the network's designated reference clock, known as the grandmaster clock, to within a desired level of precision. The grandmaster clock in a given gPTP domain is chosen using the Best Master Clock Algorithm (BMCA), which is outlined in IEEE 1588 [11]. The grandmaster clock forms the root of a spanning tree of master-slave connections between all time-aware systems on a network. 802.1AS is capable of time synchronization via 802.3 Ethernet, 802.11 WLAN, and EPON [23], however in this thesis only the Ethernet based case will be discussed. Each link from time-aware system to system is via a point-to-point physical Ethernet connection to an 802.1AS capable port on the device. Each port on a time-aware system can assume one of four states, as defined below in [23]:

- **Master Port**: Any port of a time-aware system which is the closest to the root from the view of a subsequent system in a spanning tree. The synchronization messages are transmitted along the spanning tree.
- **Slave Port**: One port of a time-aware system which is the closest to the root of the spanning tree. At this port, synchronization messages are received. The grandmaster does not have assigned slave ports.
- **Disabled port**: Any port of the time-aware system which is not IEEE 802.1AS capable.
- **Passive port**: Any port of the time aware system which is neither master port or slave port or disabled port.

Synchronization messages originate at the master ports of the grandmaster, and are disseminated from the root to the slave ports of the subsequent layer of time-aware systems. These systems then send synchronization messages from their own master ports to slave ports at the next hop in the network, and so on. Figure 2.1 shows a simple network as described.

Mechanism of Synchronization

At the core of the 802.1AS synchronization mechanism is the sending and receiving of synchronization messages with highly precise hardware and information

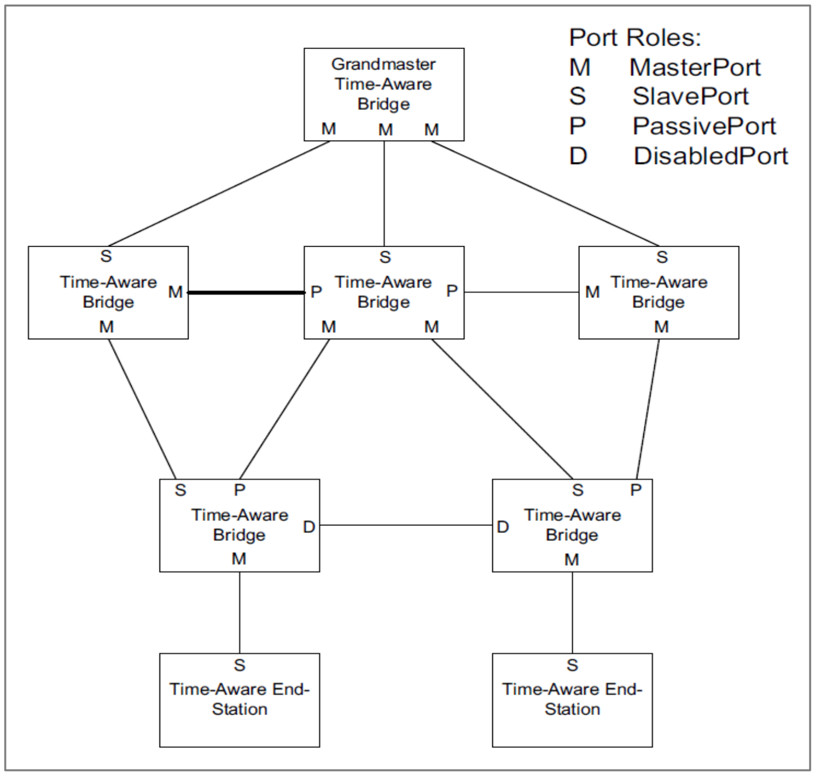


Figure 2.1: Distribution of port types in an 802.1AS Network [8].

used in the synchronization process. This information is then interpreted by each time-aware system, as to enable periodic correction of its internal clock to the clock of the grandmaster. Strict adherence to ideal real-time is of less concern than strict adherence to the grandmaster's clock, as even the grandmaster clock is not ideal. The synchronization messages (or *Sync* messages) from master to slave contain the following timing information used for synchronization [7]:

- **Precise Origin Timestamp**: Highly precise timestamp when the current *Sync* message was transmitted from the master. This information is only included in the S*ync* message when using one-step mode, otherwise it is sent in a *follow-up* message (two-step mode).
- **Rate Ratio**: The ratio of the frequency of the grandmaster clock to the frequency of the local clock. The current sending master (i) calculates this value based on information received in the previous upstream *Sync* message from the previous

master (i-1), the rate ratio of the neighbor system (i-1) that sent the upstream *Sync* message.

- **Correction Field**: The time that it takes for *Sync* messages to reach the grandmaster to the current (i-th) time-aware system. It is a sum of all the previous physical propagation delay times as well as the residence (processing) times in the previous time-aware bridges or systems.

The i-th time-aware system, knowing the precise time that the initial series of *Sync* messages left the grandmaster, as well as the sum total of all propagation delay times and residence times in the previous time-aware systems, allows the current system to accurately update its internal clock, and prepare updated rate ratio and correction field values for the next downstream S*ync* message.

IEEE 802.1AS synchronizes networked clocks by sending periodic *Sync* messages from masters to slaves. Below is a diagram with noted terminology that will be required for the detailed overview of the mechanisms by which time-aware system clocks are synchronized. From Figure 2.2, the following definitions can be obtained [4]:

a) **Slave offset**: The time offset (measured in nanoseconds) of the slave clock with respect to the grandmaster clock. This value is calculated, and used to correct the clock.
b) **Peer delay**: The measured propagation delay along the physical connection between master and slave ports between two time-aware systems. This is measured in nanoseconds.
c) **Residence time**: The time (measured in nanoseconds) that a *Sync* message spends in the switch before being forwarded. This value is not static, and changes from switch to switch with changing network conditions.
d) **Sync interval**: The interval between successive *Sync* messages. The default is 125ms.
e) **Peer delay interval**: The time between two successive *Pdelay_Req* messages.

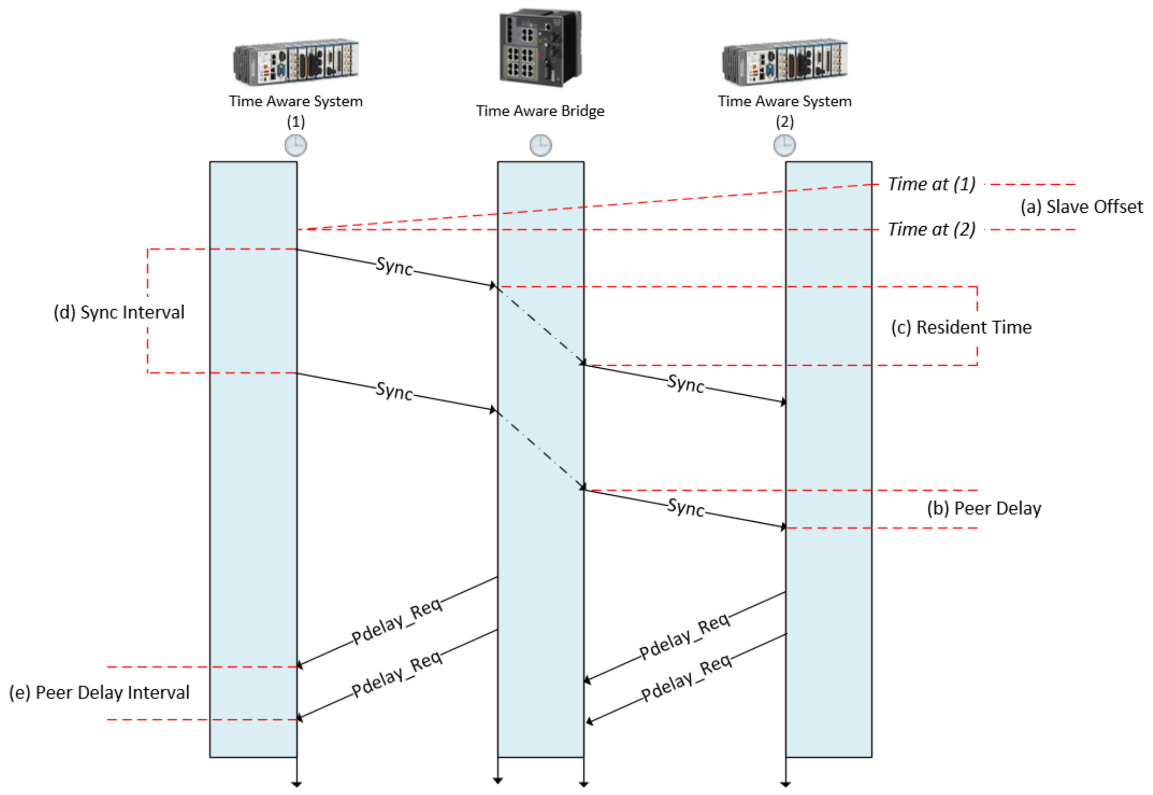


Figure 2.2: Synchronization Terminology [4].

There are several message types that are exchanged between time-aware systems, including *Sync* messages (used to pass the grandmaster reference time to all slave nodes), and *Pdelay_Req/Pdelay_Resp* messages (used to calculate path delay between peers) [21]. Additional messages used by IEEE 802.1AS and IEEE 1588 may be found in the respective standards [11, 21].

In addition to the different message types used by 802.1AS, it is important to understand the different clock types encountered in the standard. The three types of clocks associated with the protocol are: *ordinary clocks*, *transparent clocks*, and *boundary clocks* [4]. An o*rdinary clock* can be one of the following three types:

- Grandmaster clock: always acts as a master, never as a slave. Must have an accurate and precise source of time.

- Master/Slave: May act as a master clock or slave clock, depending on the needs of the network. This clock would act as a slave if the network already has a master clock with superior timing abilities (such as a grandmaster with GPS reference).
- Slave only: this type of clock acts only as a slave, never as a master.

Ordinary clocks are the types of clocks found in the end-devices on a TSN network. In contrast, transparent and boundary clocks are the types of clocks found in time-aware bridges (switches). They differ in operation somewhat compared to an ordinary clock.

- T*ransparent clock:* passes individual *Sync* messages through itself and on to their destination. However, the transparent clock updates the timestamps of the correction field of the *Sync* message, to account for forwarding delays in its switching operation [4]. A diagram of a transparent clock may be seen in Figure 2.3.
- *Boundary clock:* does not forward received *Sync* messages like a transparent clock. Rather, a boundary clock has a dedicated slave port that receives upstream *Sync* messages. The boundary clock uses the timing information to update its own clock, and then generates new unique *Sync* messages to pass downstream on its master ports [4]. See Figure 2.4 for a diagram of a boundary clock.

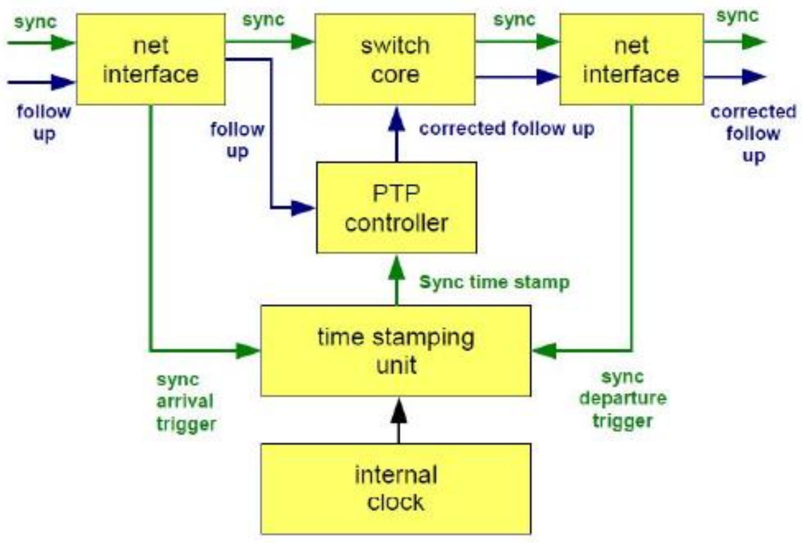


Figure 2.3: Transparent Clock [4].

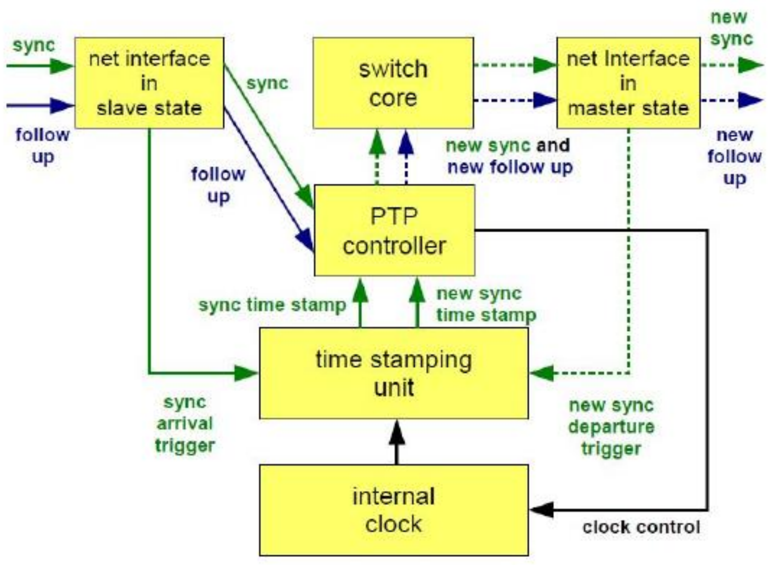


Figure 2.4: Boundary Clock [4].

As shown in Figure 2.5, multiple messages are sent between peers to establish four timestamps, used to deduce the network delay between two links. IEEE 802.1AS determines the synchronization correction information from each hop in the spanning tree, rather than from endpoint to endpoint [7]. Periodically, each time-aware system will send out a *Sync* message on its master port, containing a precise origin timestamp, rate ratio information, and correction field [7]. The rate ratio r_i is the ratio of the frequency of the grandmaster clock to the frequency of the local clock. This allows the receiving host to calculate its own rate ratio, to be passed down the chain and used in downstream calculations. IEEE 802.1AS does not describe the algorithm used to determine this value, only specifying that it must be measured to within 0.1 ppm [7].

The correction field C_i is the time that it takes for the synchronization information to make it from the master clock of the i'th node in the chain, accumulating delay after each hop. This includes both propagation time as well as residence time. To determine the propagation delay, a series of messages are sent from node i to node i-1, to generate four timestamps to be used in the calculation of propagation delay and residence time of each link. It should be noted that while IEEE 802.1AS has mechanisms in place to compensate for known non-symmetrical links, it cannot compensate for these asymmetries on the fly [7].

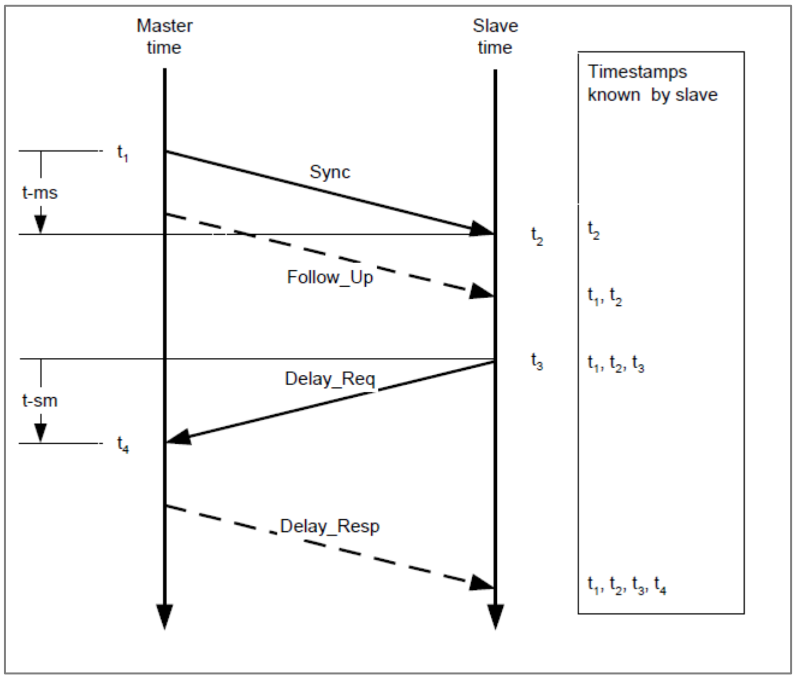


Figure 2.5: Two step Sync and propagation delay measurement [11].

There exist two synchronization update modes in IEEE 802.1AS, which are selected via the *syncLocked* flag. If this flag is true, every time-aware system will send a downstream *Sync* message upon the receipt of an upstream *Sync* message. If this flag is false, each time-aware system sends *Sync* messages at some periodic interval which is independent of the receipt of its own *Sync* messages on its slave port. While the first case allows for higher levels of synchronization precision, it leads to higher bandwidth. The second case reduces precision somewhat, but allows for a more deterministic network system setup [7].

Due to the granularity of system clocks in time-aware systems, in which clocks are generally only accurate to single digit nanoseconds, the delay time calculated from timestamps will be necessarily truncated to the nearest clock cycle of the time-aware system [7]. In addition, variances in the delay of the signals while traversing the physical layer (known as PHY jitter) also results in variance in the accuracy of peer delay measurements. Both of these physical phenomenon together can lead to large inaccuracies in measurement and correction, reducing time precision. This can be resolved with the addition of an averaging filter (shown in Figure 2.6), as these clock

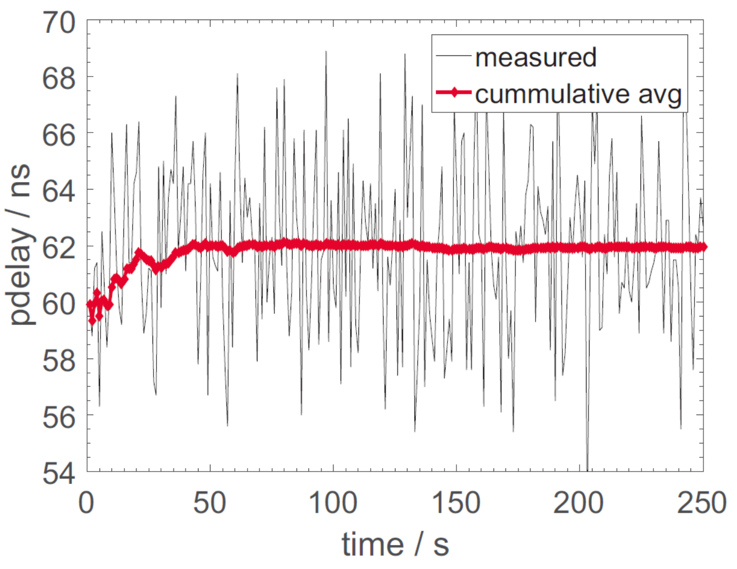


Figure 2.6: Delay measurements caused by limited clock resolution can be improved by use of an averaging filter [7].

truncation and PHY jitter errors will, on average over many samples, cancel out. With a filter in place, average measured delay value approaches D_i, which is the sum of the physical delay D_0, 2J (J being the average PHY jitter), and g (the clock granularity, or time between two consecutive clock ticks) [7]. This improves the precision considerably [7, 23].

Synchronization Considerations and Performance

Typically for an industrial application, it is desirable to achieve clock synchronization to within 1µs [7]. Generally, the most important aspect of synchronization is not the absolute accuracy of the of the system clocks to "real" time, but the precision of the clocks amongst themselves and the grandmaster. The "precision" of the clocks in a time-aware network is defined in [7] as "the maximum difference of the local clocks in the network of non-faulty nodes at any point in time during the operation of the system". In other words, generally clocks need to be synced with each other, rather than being synchronized to the real world.

802.1AS specifies that it can maintain a precision of 1µs up to 6 "hops" from the grandmaster [21]. Various tests in literature demonstrate however that IEEE 802.1AS can achieve much greater levels of precision for small systems, or acceptable levels of

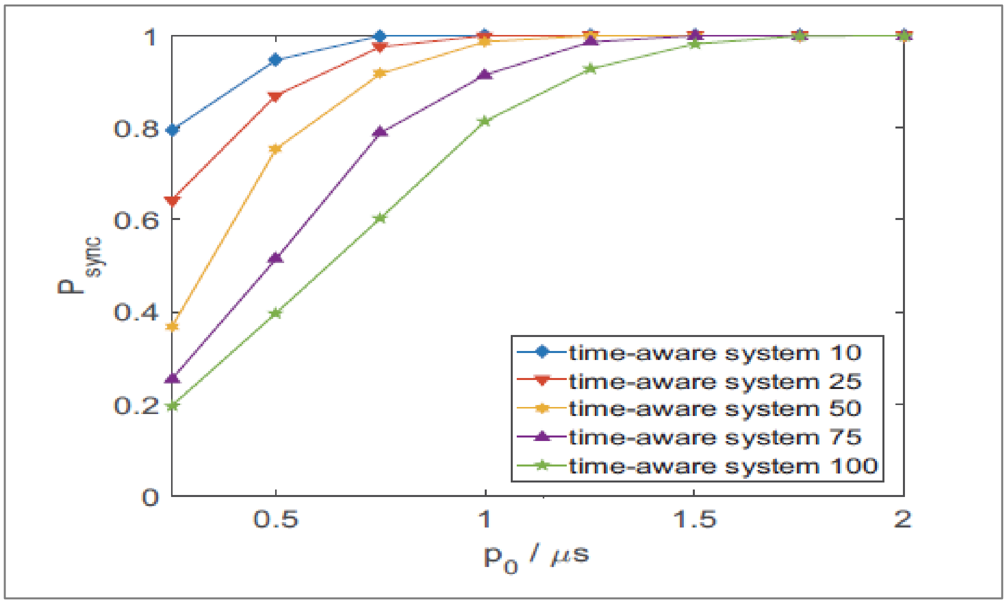


Figure 2.7: Probability of synchronization within a given precision [7].

precision for larger systems. For example, M. Gutiérrez et.al. demonstrates that assuming worst case conditions, a 100 hop 802.1AS system can maintain a synchronization precision of 6.3µs [7]. However, this assumes that all errors added together constructively, whereas in the real world random errors tend to cancel out over larger samples. When probable error values were allowed to be truly random, the synchronization precision for a 100 hop system was reduced to +/- 2µs, and precision of <1µs was maintainable for systems with up to 30 hops [7]. The probability that a given level of precision can be maintained for different system sizes is shown in Figure 2.7.

Given the branching nature of Ethernet networks with time-aware bridges, enormous networks of time-aware systems are theoretically possible. For example, a branching network of 12-port time-aware bridges with a maximum of ten hops accommodates tens of millions of endpoints. Other literature has shown that in very small networks of 2-4 hops, synchronization precision of 100-200ns is possible, independent of high levels of network congestion [4, 23]. The preliminary experiments of Chapter 3 verify clock precision <1µs.

In addition to the features and mechanism specified for 802.1AS, 802.1AS-Rev includes the addition of several enhancements over its predecessor, including [2]:

- Explicit (rather than implicit) support for One-Step processing. Two-step processing is still supported.
- Multiple Grandmaster clocks are supported, with one as primary and another as a “hot-standby”.
- Support for multiple paths for synchronization propagation, with metrics to determine path quality.

2.1.2 IEEE 802.1Qbv – Time Aware Traffic Shaping

IEEE 802.1Qbv, known as “time aware traffic shaping”, is the standard that governs scheduling different flows through a TSN network to guarantee deterministic latency intended to meet requirements for different data priority levels. It is an amendment to the existing IEEE 802.1Q-2014 standard, which governs virtual LANs (VLANS) on an IEEE 802.3 Ethernet network [24]. This system, which is adapted from the AVB 802.1Qav standard, was inspired by the *time slot procedure* implemented by PROFINET IRT [12]. Unlike 802.1Qav, where endpoints performed the traffic scheduling, using a *credit based shaper*, in 802.1Qbv the switches become the most important part of the traffic shaping process [12].

The 802.1Qbv standard allows certain high priority traffic to make its way through a network within a maximum allowable latency, regardless of other high bandwidth traffic present. This standard is heavily dependent on the previously discussed IEEE 802.1AS, as 802.1Qbv uses the time synchronization features provided by 802.1AS to follow a strict flow schedule [2]. This flow schedule determines certain windows during a specified period for the time-aware systems throughout a TSN network to block

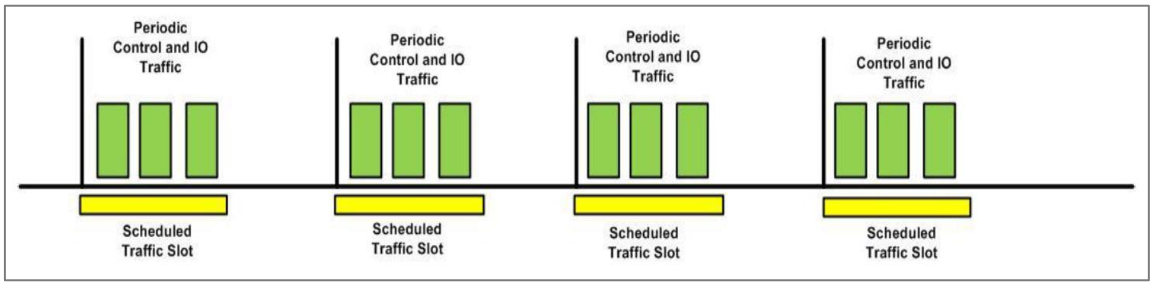

Periodic Control and IO Traffic
Periodic Control and IO Traffic
Periodic Control and IO Traffic
Periodic Control and IO Traffic
Scheduled Traffic Slot
Scheduled Traffic Slot
Scheduled Traffic Slot
Scheduled Traffic Slot

Figure 2.8: IEEE 802.1Qbv Traffic Slots for Priority Traffic [2].

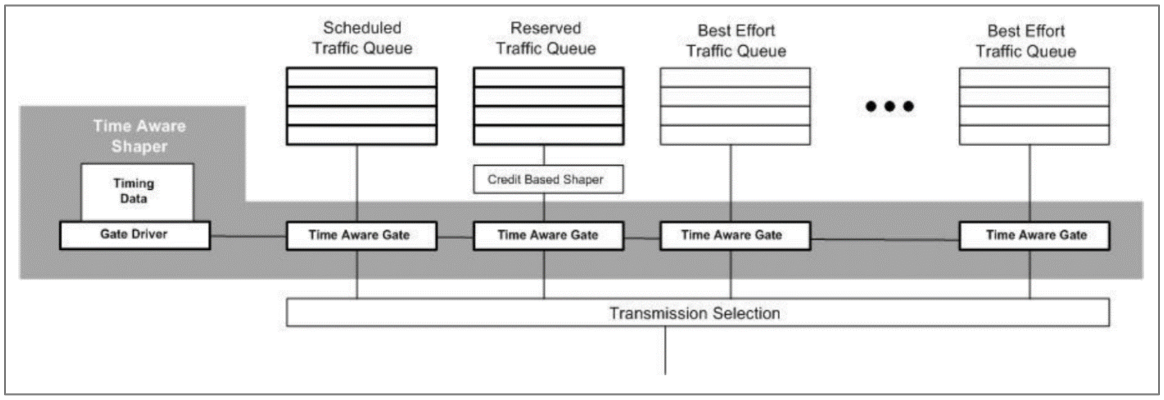


Figure 2.9: Time Aware Shaping Queues in IEEE 802.1Qbv [2].

all other traffic except high priority traffic, allowing it to navigate a system swiftly and without delay. Ensuring all bridges adhere to this schedule requires them all to have a common sense of time. The bridges of the system can be thought of as traffic police, letting specified packets flow freely within a certain period during every cycle of the schedule [24]. The standard has been designed, like 802.1Qav, to handle roughly 75% of a network's traffic requiring strict latency requirements, though specific settings change the exact proportion [2, 3].

The features provided by 802.1Qbv are highly beneficial to systems that require near real time control or monitoring capabilities, as latency can be reduced consistently to one millisecond or less. This is beneficial for applications where tight close loop control is required, such as industrial applications and robotics [24].

Mechanism of Operation

The basic mechanism behind the traffic shaping features of 802.1Qbv is the tightly scheduled sending of data into the network. This is facilitated by the time-aware bridges of a TSN network by the internal time aware shaper present in the switches. This component can be thought of as a gatekeeper, allowing certain traffic to "skip the line" during a designated window of each time period. The shaper knows at what point to stop other traffic due to the schedule received from the network Central Network Controller (CNC, discussed in 2.1.4), and the common perception of time experienced by all network switches due to 802.1AS. At some point in the schedule, the switches know to stop all traffic flow except the designated traffic, allowing this data to make it to its destination at the fastest possible speed. Figure 2.9 shows a representation of the time aware shaper, which opens and closes gates to allow different levels of traffic into the network according to the set schedule. Figure 2.8 shows a representation of the data frames being periodically sent during the designated portion of the set period.

2.1.3 IEEE 802.1Qcc – Stream Reservation and Central Management

802.1Qcc outlines the central user management and network configuration aspects of TSN. 802.1Qcc has less to do with the function of the standard and more to do with the control, configuration, and user interface of a TSN system and application. To this end, the standard (in draft form at the time of this writing) specifies the following as enhancements [10]:

- Specifies a software interface between the TSN application and the physical network components. The user inputs requirements, and the network configures itself to meet the requirements accordingly. Specified as an "information model", in that it can be applied to any protocol.
- Specifies three models for the user/network interface (UNI): fully centralized, fully distributed, and centralized network/distributed user
- Specifies various enhancements to the Stream Reservation Protocol (SRP)
- Specifies enhancements to the managed objects for Forwarding and Queuing Enhancements for Time-Sensitive Streams (FQTSS)
- Details enhancements to the SRP managed objects
- Specifies managed objects for switch configuration by a Centralized Network Controller (CNC).

In essence, 802.1Qcc focuses on specifying standards for systems to control and configure TSN network administration. Figure 2.10 shows a basic network system demonstrating the centralized control outlined in 802.1Qcc necessary for a TSN network. This is the scheme that the Cisco and National Instruments equipment used in the tests discussed in Chapter 3 provide for administration. This model (as described in the

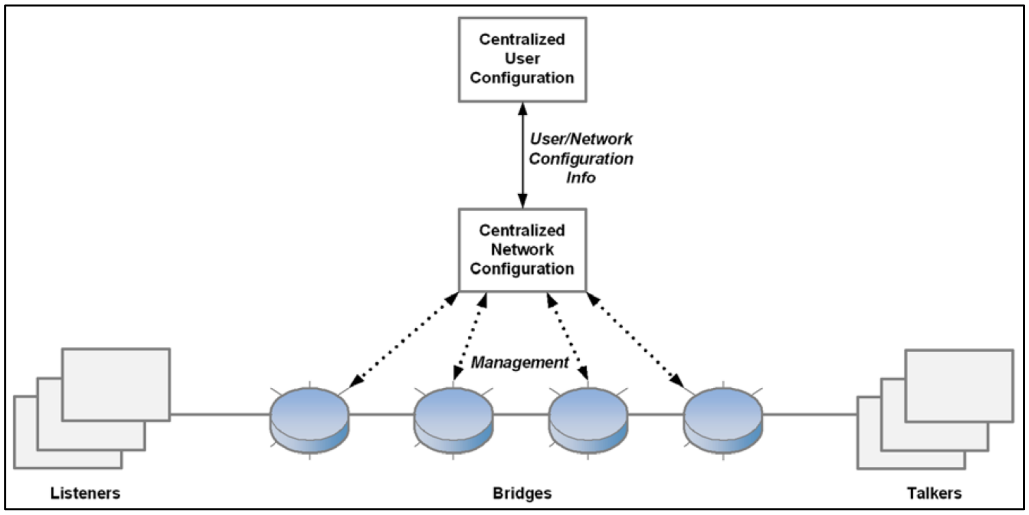

Centralized User Configuration
User/Network Configuration Info
Centralized Network Configuration
Management
Listeners
Bridges
Talkers

Figure 2.10: Centralized Control Model [10].

standard) is more apt for applications that require a large degree of user configuration of the end stations in a system [10]. Many industrial applications that could utilize TSN are heavily dependent on the physical environment and other equipment required in the application, and thus require a higher degree of timing and coordination on a network wide level. The centralized approach makes configuration of this sort of network easier than a distributed approach. In this approach, all TSN administration is done via the CNC, including specifying from which talker to which listener(s) are part of a particular flow, as well as timing and frame size settings. These settings are them pushed from the CNC to the TAB after the TSN network schedule has been calculated. The Cisco TSN switches then administer the network as configured via the CNC.

In contrast, a distributed approach (shown in Figure 2.11) may not meet the network requirements as a whole, as each bridge is limited in the knowledge of the entire network due to the propagation of information inherent in that approach. This method utilizes the TSN links themselves to propagate information through the network, as requested by users (talkers/listeners) at the endpoints. There is no centralized control in a distributed model. In addition, a Centralized Network/Distributed User model (shown in Figure 2.12) may not provide the necessary level of control required for more complex applications, despite the centralized nature of network configuration [10]. In this model, the requirements are still specified by the users at endpoints, but this information is communicated to a central authority, which makes scheduling and routing decision with the entire network in mind [10].

2.1.4 Additional TSN Standards

In addition to the standards reviewed in the previous section, there exist several other TSN standards, some published and some in draft stages, which are not featured on the hardware used in this implementation, or were not available on commercial products at the beginning of this project. However, a new switch family available from Broadcom includes compatibly with the TSN standards described in this section, and more commercial products featuring the full suite of TSN standards will surely follow.

IEEE 802.1Qbu – Frame Preemption

IEEE 802.1Qbu Frame Preemption specifies methods to split Ethernet packets that intrude into the "high priority" transmit time of a TSN implementation. Currently, a packet that incurs upon the time slot designated for high priority traffic is lost (as the

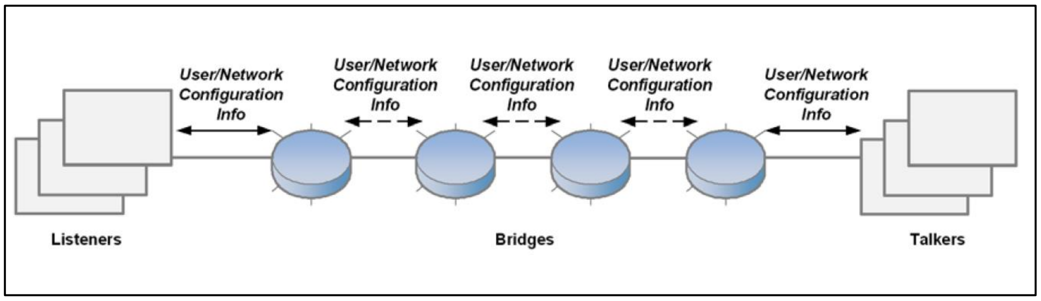

User/Network Configuration Info
User/Network Configuration Info
User/Network Configuration Info
User/Network Configuration Info
User/Network Configuration Info
Listeners
Bridges
Talkers

Figure 2.11: Fully Distributed Model [10].

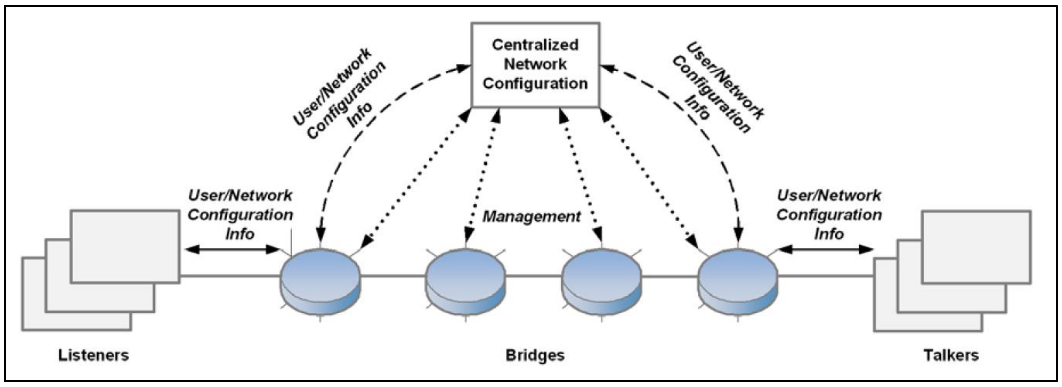

Centralized Network Configuration
User/Network Configuration Info
User/Network Configuration Info
User/Network Configuration Info
Management
User/Network Configuration Info
Listeners
Bridges
Talkers

Figure 2.12: Centralized Network/Distributed User Model [10].

transmission is ceased and not resumed) [2]. In order to circumvent this, higher order protocols with built in error handing for lost packets (such as TCP) would detect this and request a retransmit. However, 802.1Qbu specifies methods by which the packet that spills into the high priority scheduled slot is held in the time-aware bridge until the conclusion of the time slot, and then transmission of this packet continues as before. This technology allows for less network bandwidth and delay due to dropping best-effort traffic to make way for TSN packets. This standard has been published, but is not currently available (at the time of this writing) on commercial hardware. An example of frame preemption can be seen in Figure 2.13.

IEEE 802.1Qca – Path Control and Reservation

Path control and reservation, as the name implies, governs the reservation and control of paths through a TSN network. This means that an administrator can set specific forwarding rules for specific types of traffic. For example, a rule might be set mandating that all blue traffic must take route X, while all red traffic must take route Y . This could be used to prevent conflicting data types from sharing the same network path, despite a

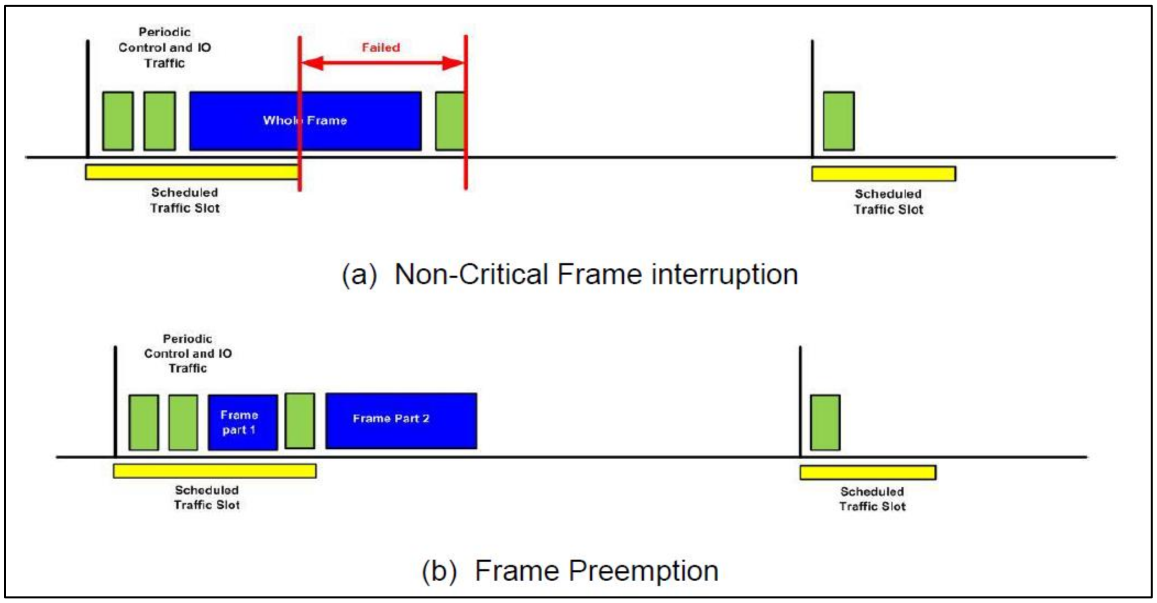


Figure 2.13: Frame preemption and interspersing express traffic [2].

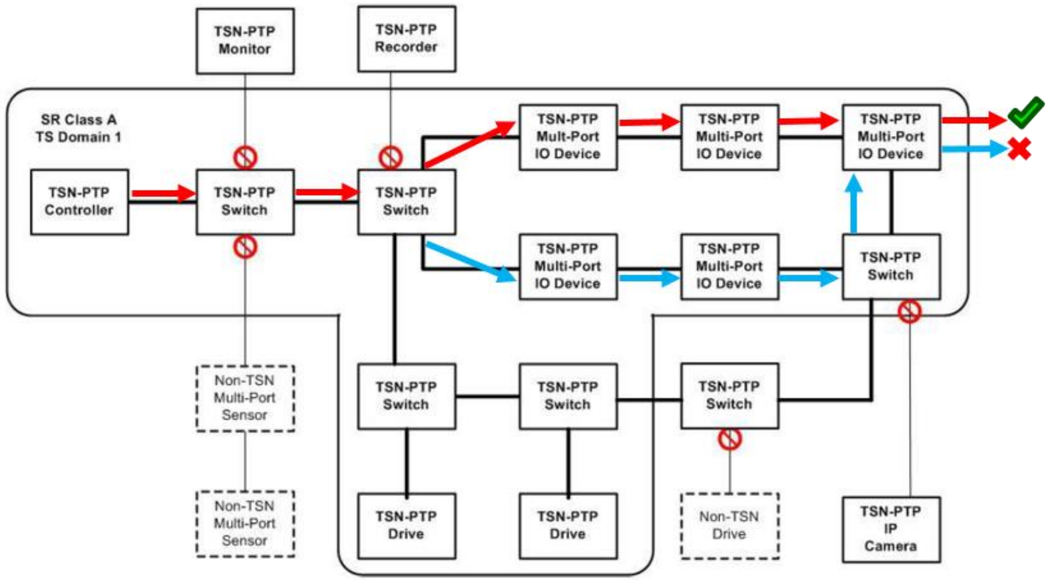


Figure 2.14: IEEE 802.1CB allows duplicate frames to be sent along different paths, and eliminates redundant frames at the destination [2].

standard switching configuration that would lead to this ("Shortest Path Bridging" [2]). In addition, tools for reserving set bandwidth or streams are also defined in 802.1Qca, as well as redundancy features. This standard is closely related to 802.1Qcc and 802.1CB, as all three fall under the Stream Reservation Protocol (SRP) functional group.

IEEE 802.1CB – Frame Replication and Elimination for Reliability

This standard governs improved reliability features provided to a TSN network. IEEE 802.1CB outlines mechanisms to send duplicate data frames over different network paths for redundancy, and then discard unnecessary duplicate packets at the destination [2]. The duplicate frames can be created in either the end stations, or along the path at bridges. An additional standard that governs a similar method for duplicate frame generation and elimination is defined in IEC 62439-3 [2].

IEEE 802.1Qci – Per-stream Filtering and Policing

This standard specifies methods and mechanisms to enable filtering, counting, and policing of Ethernet frames based on TSN data stream information [2]. 802.1Qci also has mechanisms to have these features run on a synchronized schedule. The focus of the standard is to use the filtering and policing strategies to mitigate undesirable network traffic/transmission, to improve overall network performance [2]. Also, the standard specifies time-based ingress policing, in which certain frames can enter or exit the next network node/element only at certain times based on a schedule (similar to 802.1Qbv). This improves network determinism, as well as provides some potential security considerations (as packets received outside of the scheduled time period are dropped). The standard is an amendment to the 802.1Q standard, governing virtual LANs (VLANS). In addition, the standard is closely related to both 802.1Qcc and 802.1CB, as it focuses on the management features of TSN.

2.2 AREAS FOR POTENTIAL TSN INTEGRATION

This section will outline various ways in which the main features of the Time Sensitive Networking suite of standards discussed in this implementation (time synchronization and deterministic data flows) can be used in microgrid and smart grid power systems. Many systems discussed in literature assume minimal or no high bandwidth communication links, due to concerns with cost, reliability, modularity, and expandability [16, 18, 25]. For many applications, distributed solutions or minimal low bandwidth links are good enough, and larger degrees of control do not justify these concerns. However, the features and benefits of TSN discussed here will necessarily assume that an Ethernet backbone is in place. The Ethernet standard (with TSN improvements) can dramatically improve the drawbacks of previous communication systems and appease the concerns about reliability, modularity, and expandability, though higher costs (when compared with a system with minimal or no communication links) is still a factor. It would be up to the power system operators to determine if the benefits and potential cost savings of a TSN Ethernet based communication network would justify the expense over a less costly communication scheme.

Most microgrid scenarios discussed in the following sections will assume the centralized approach, as it provides better global optimization that can benefit from the features of TSN. Distributed and decentralized systems can also benefit to some degree from TSN, but will not be the focus of the benefits presented.

2.2.1 Microgrid and Distributed Generation Control

As more and more distributed generation resources, such as solar and wind resources, come online, the need for a scalable system to control and monitor these resources becomes paramount. This is especially true for residential PV deployments, where there are many small entities that all must act together. Recent legislation in California has even gone so far as to mandate that every new house must include solar generation by 2020 [26], further demonstrating the current need for large scale coordination of many small residential PV deployments. In addition, data from the National Renewable Energy Laboratory (NREL) shows that solar energy adoptions continue to surge, while prices continue to fall [3]. A scalable system to control all of these deployments (in the form of a microgrid) is one area where features of IEEE 802.1 TSN should be explored. The deterministic data flow features of TSN can allow for scalability of an Ethernet based control network to grow with the number of PV deployments.

Microgrid Control Hierarchy

Microgrid control is generally divided into three layers: primary, secondary and tertiary [17, 19, 27]. Primary control is implemented locally at each inverter or distributed generation node, and is often accomplished through various droop control mechanisms [17, 28]. In a droop control scheme, an increase in load power causes an increase in generation power, with a corresponding reduction in frequency [27]. Due to inherent limitations in droop control, voltage and frequency tend to vary from the desired set points due to accumulating steady state errors. This is due to the fact that the change in the generated power is smaller than the increase in load power, leading to the accumulating errors in frequency [27]. Primary control is independent, not requiring outside control or communication functionality [19]. Communication generally becomes necessary to effectively implement the secondary control level.

Secondary control compensates for this error and ensures that frequency and voltage remain at their desired values. The nominal primary setpoint frequency f_{nom} is regulated using the secondary control methods, to ensure that the actual frequency output by the inverter matches the desired setpoint [27]. This value is regulated based on grid or microgrid reference measurements, or some other method. Secondary control is often centrally controlled over the entire microgrid, thus requiring a communication backbone to deliver the commands to distributed generation resources [19]. Despite central control's superior performance, high cost of communication links and reliability concerns have led to research into methods of secondary control that do not rely on high bandwidth communication [15-17, 19, 28, 29]. In addition to frequency regulation, supplemental control functions such as voltage unbalance and harmonic compensation can be included as part of the secondary control loops [19].

Tertiary control, unlike primary and secondary control, works on the order of days or weeks instead of seconds or minutes. Tertiary control involves large scale economics driven optimizations of the microgrid in the context of the main grid power system, and involves longer timescales and power flow [27].

When speaking of microgrid control, the control action required is to achieve frequency and voltage regulation, active and reactive power balance between DERs, main grid resynchronization, energy management, and optimizing microgrid performance to maximize economic considerations [19]. In addition, control information such as set-points, timing information, and orders from grid operators must be communicated quickly and reliability to the distributed resources to effectively administer the microgrid. In a microgrid, generally each inverter would be accompanied by a local controller, which interfaces between the inverter and the microgrid communication/control plane. The local controller is responsible for the voltage and frequency control of an inverter based on the control mode of the entire microgrid (islanded, grid-connected, black start, transition, etc.) [14]. Control is fairly straightforward during grid connected mode, as the inverter uses PQ control modes to match phase and frequency with the strong grid reference available [14]. However, when a microgrid is islanding, there is no strong reference, and inverters must assume Voltage/frequency (V/f) control mode, which requires a voltage set-point and frequency reference [14].

In the majority of microgrid related publications focusing on control, the focus is to avoid central control topologies if possible [15, 17, 18, 29]. The motivation behind decentralized and distributed microgrid control strategies over centralized is the inherent need of an extensive high bandwidth communication system for a centralized approach, for which the cost and reliability is of some concern [15, 17, 27]. Despite the higher levels of control and optimization possible in a centralized approach, there is significant research into gaining greater levels of control in the non-centralized topologies. Some novel methods proposed in research include systems that only require local controllers to communicate to their nearest neighbor, eliminating the need for many high bandwidth lines [17, 27]. Still other research focuses on hybrid approaches, where there is some local controller autonomy to make decisions without communication overhead, while a central control still coordinates long timescale settings and control using a lower bandwidth method of communication.

As part of a distributed secondary control scheme discussed in [19], latency tests were conducted to see how higher delays affected centralized systems vs a novel distributed system. It was shown that a centralized system is less able to achieve acceptable performance at higher levels of delay and data loss when compared to a distributed system. This was demonstrated with induced latencies of 200 ms, 1 s, and 2 s. These latencies were frequently observed (and exceeded) in high congestion network tests presented in Chapter 3, showing that in an Ethernet based TCP network, high latency is a very real possibility. However, some centralized intensity is still required for coordinated microgrid functions such as black start or global microgrid optimization [19]. Previous research has shown that communication based central secondary control can keep good performance up to latencies of 200 ms [25].

Still other research has been done into embracing communication links between controllers, but utilizing methods to detect and mitigate the drawbacks of traditional methods, such as latency and clock drift [27, 30]. While these methods show promise, they rely on the identification of communication link inhibitions, rather than the reduction of the inhibitions. This is contrast to IEEE TSN, in which mechanisms actually reduce the latency experienced, rather than measure and compensate for it.

TSN alleviates some of this concern due to its ability to create redundant frames along separate paths, its ability to segregate different traffic into different paths, and its deterministic latency features guaranteeing fast and reliable control and measurement delivery. A central controller is the best method to coordinate fast responses and global optimal microgrid states, as it is impossible to achieve this condition with each DER acting independently [18, 27]. The potential for a microgrid system to have the fast response of a decentralized approach, while also having the global optimization capability of a centralized approach, is one benefit of using a TSN based communication and control system. This system would allow for the rapid dissemination of information through the grid, while also allowing for relatively easy plug and play expansion of the communication network due to its Ethernet base. In addition, the rapid dissemination of information with minimal latency through the microgrid could allow for the use of more optimal control algorithms that fail to work when higher latency is introduced. One example of this can be found in [31], where a promising control algorithm was found to have unacceptable performance with the introduction of a 500 ms communication delay. In addition, even acceptable control algorithms can be shown to converge more slowly or lead to less optimal results when paired with higher communication latencies [32].

Microgrid Communications

Currently, power systems and microgrids use various means to implement communication networks for control and monitoring. Physically, these systems can operate with connections based on fiber, copper, or transmission line (PLC) links [33]. If physical links are not possible, wireless links (GPS, wireless LAN, radio) systems have been used. Some common methods use GPS to provide a common time reference across a network, while using a sparse communication system for low bandwidth setpoints in conjunction.

Standard LAN is already generally thought to be acceptable for communication systems that are semi-autonomous, in that they do not require constant communication to function [6]. Some small microgrid testbeds or single building systems have been run using a LAN based system, using a TCP/IP based protocol, but systems such as these may not scale efficiently as the communication network becomes larger and more distributed [25]. In an attempt to improve performance of LAN networks in industrial applications and SCADA systems, the Enhanced Performance Architecture (EPA) is

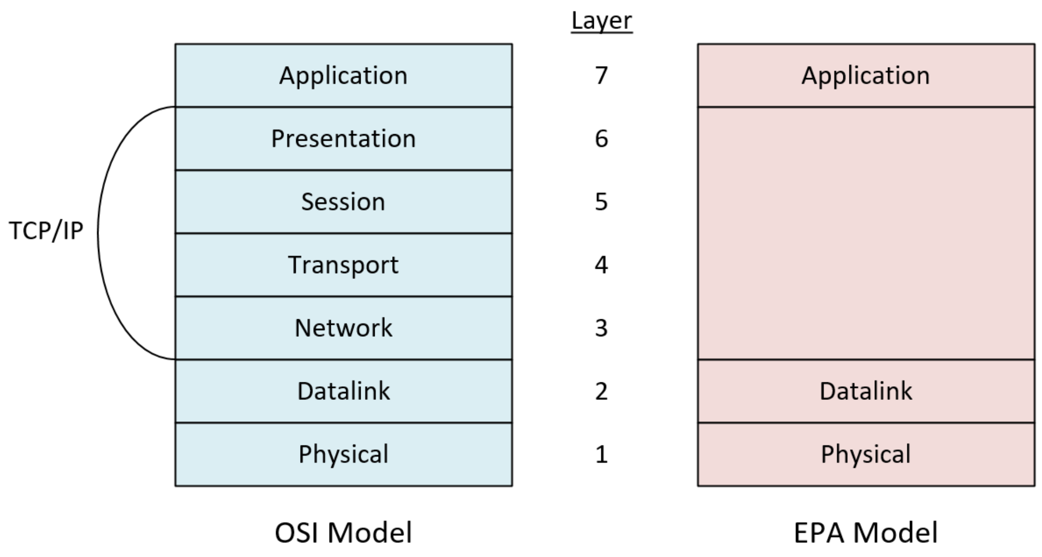


Figure 2.15: Comparison of the OSI communications model with the EPA model [6].

often used as the communication architecture, as opposed to the standard OSI model [6]. The EPA model uses only layers 1, 2, and 7 of the OSI model, improving performance at the expense of functionality. In the EPA model, data is directly input into the datalink layer from the application layer, bypassing many of the other layers in between, reducing overhead and improving speed. This is the method used in the TSN implementation of this thesis, as TSN is a Layer 2 datalink layer standard being configured by LabVIEW (a layer 7 application). TSN is compared to UDP and TCP in Chapters 3 and 4, which are found on layer 4 of the OSI model, and are better suited for speed and reliability, respectively. The EPA and OSI models are shown in Figure 2.15.

According to [34], IP based communication networks are already satisfactory for smart grid applications. However, calculations show that to achieve acceptable levels of latency (10ms), only approximately 10% of the communication network capacity can be used at any time, else latency will increase to unacceptable levels [34]. (Based on 10e6 100bit messages per second, with 99% reaching their destination in under 10ms). This is very poor bandwidth utilization. TSN based LAN networks allow for higher levels of bandwidth utilization (75%) with lower latencies [2, 3].

Power line communications (PLC) is another option for low bandwidth communications. PLC induces ripples in the distribution line itself to send data to receivers at other stations around a service area [33]. PLC induces a 3 kHz to 95 kHz ripple on top of the line frequency for bidirectional communication. However, the low bandwidth capacity of PLC limits its practically as a microgrid communication method [35]. Research has been conducted into improving the data rate of this communication method, but the upper limit of data rates is limited due to the signal attenuation at the higher frequencies required for higher bandwidth transmission. This could be circumvented using repeaters, which is an expensive and undesirable solution [33]. In addition, PLC systems are more prone to interference from inverter switching, making them less ideal for microgrid systems made up of predominately switching inverters.

Higher levels of microgrid interconnectivity and data logging also allows for power utilities to improve system automation [33]. This can lead to reduced operation costs and improved system utilization of available distributed resources and more efficient use of power globally through the microgrid [33].

One important aspect of any microgrid communication system is that it needs to be scalable. This includes not only the physical scalability of the hardware capability, but also the scalability of the administration and configuration systems in place as more and more nodes are included on a network [33]. In a large scale system communication system for large arrays of distributed residential PV, the need for some level of automation will become inevitable. The Ethernet standard that IEEE TSN is built on provides for these concerns, as it is already proven in enterprise as a scalable, configurable system that can be automated with the right tools. In addition, since TSN runs on standard Ethernet, many of the best-practices standard to large network administration of Ethernet based networks carry over to TSN Ethernet based networks. This includes the use of firewalls for security, the ability to administer VLANs, encryption of streams, SSL/TLS, secure certificate authentication, etc. Also, as a subset of standard Ethernet, TSN features are both backwards compatible with non-TSN Ethernet based hardware (without using TSN features), and will continue to benefit from improvements to the Ethernet standard in the future.

2.2.2 Microgrid Inverter Synchronization

In addition to the higher levels of control capability afforded to a power system utilizing TSN Ethernet system for communication, another benefit is the ability to utilize the timing and synchronization capabilities to facilitate inverter synchronization in a microgrid. High degrees of grid timing precision can be used to bring all distributed sources online at the same time in a synchronized fashion, or synchronize an entire microgrid to the main grid for reconnection. Tight synchronization between distributed energy resources in a microgrid is incredibly important, as relatively small circulating currents caused by a mismatch can exceed the ratings of the distributed resources [36].

Microgrids must function in voltage control mode during islanded operation (to control frequency and voltage) and transition to current control mode during grid-connected operation [30]. The main power grid acts as a “stiff” reference to the inverter to synchronize its out output. The grid is a “high-inertia power system”, consisting of many spinning generators, each with physical inertia in the form of spinning turbines. This inertia allows them to respond to changes in load rapidly through physical means, by speeding up or slowing down in response to load changes [36, 37]. A system comprised predominantly of power electronic based sources (such as a microgrid with many DER connected inverters), has little to no system inertia.

In an islanded microgrid scenario with many small inverters and low system inertia, the grid formed along the microgrid bus is very weak, preventing precise synchronization of inverters using a method based on simply tracking the grid voltage and phase [30, 36-38]. In addition, variations in non-ideal clocks of inverters make it impossible to simply use timing information to keep multiple inverters synchronized [16, 36, 37]. In testing performed in [37], two inverters synchronized and left to run based on their system clocks began to diverge after only 80 seconds. In the presence of a strong grid, it is generally easy for inverter systems to simply track and match the main grid frequency, phase, and voltage. In a microgrid without many grid forming sources, this is not possible [36, 37]. A single grid forming source is likely too small to form a strong enough grid on its own to form a grid strong enough for synchronization by the other distributed resource inverters. Thus, synchronization of many distributed sources is required to form a grid strong enough for the remaining sources to match. Lack of adequate synchronization would lead to rapid changes in grid conditions which will quickly lead to microgrid collapse. Thus, the accepted rule is to only island when necessary to continue power flow to the loads when the main grid is unavailable.

When transitioning from grid-connected to islanded mode, the change from current control to voltage control must occur instantaneously, as to prevent circulating currents in the microgrid system [8, 30, 39]. In systems with spinning generators, out of sync breaker closures can also cause vibrations and oscillations in the shaft, shortening the lifespan and potentially tripping the generator [8]. A method by which this transition can be achieved using the determinism and synchronization features of TSN is to schedule this transition shortly in advance. The latency guarantees present in TSN would allow the control command for this transition to reach all inverters in under one electrical cycle. This control command would contain the precise time that the PCC would be opened, and all inverters should transition to voltage control mode. Since, all clocks on the TSN Ethernet network are synchronized, all inverters would transition states at the same instant as the PCC is opened.

In addition to assisting with the timing of microgrid islanding, the timing synchronization features of TSN can also be used to synchronize the inverters themselves in a grid-forming or black start scenario. There are two general approaches to grid synchronization: all at once or one at a time. This is due to the very different ratings and dynamic characteristics found in renewable energy sources such as wind and solar. The rating and variability of these sources change with unpredictable environmental conditions, which makes synchronization more difficult than in a spinning generator [8, 30]. The first approach, synchronizing all sources at once, is more difficult due to the dispersed nature of the sources, requiring some form of communication or coordination to bring them online at once. However, the second approach, adding one source at a time, takes much longer time to form a stable grid [30]. Generally speaking, the method by which the synchronization occurs involves some method to identify the frequency and phase of the main grid, be it zero crossing detection, phase-lock loop (PLL), synchronous reference frame (SFR) PLL, resonant controllers, or some other method [30, 35]. This frequency and phase information is then used to generate a sine wave template for the distributed generation sources to track in sync with the grid [30]. Generally, the communication based methods use GPS or NTP to facilitate the subsystem synchronization. Although the initial cost for installation is higher, reliable high bandwidth communication links are preferred when available in a microgrid [30]. This allows for communication assisted grid synchronization (CAGS), which is faster than other methods without these links. However, cost for GPS based systems for every node is high, and using NTP for this type of time synchronization is generally not highly precise [35, 40]. One other consideration is wireless signals susceptibility to attacks. For example, is has been shown that GPS is potentially susceptible to remote timing attacks, where GPS receivers are spoofed with counterfeit signals [41].

NTP is generally precise to single digits of milliseconds. Precision Time Protocol (PTP), of which 802.1AS is a profile, is generally precise to less than 1µs, however operating system limitations can limit this accuracy to 10s of microseconds. 802.1AS is precise to below a single microsecond [7, 21]. In [42], the timing precision of the Compact RIOs used in the experiments conducted in Chapter 3 are stated to be 100ns, however tests conducted in support of this thesis were only able to achieve 700ns consistent precision

In [30] and [43], a similar method to the one proposed in Chapter 3 of this thesis is presented, using communication assistance to aid in grid synchronization using a sync waveform. This method uses a novel approach to identify network latencies on a CAN network and account for them during synchronization. This is in contrast to an 802.1AS approach, where the timing and latency inhibitions can be effectively eliminated at the link layer, rather than the application layer.

Other communication heavy methods of inverter synchronization that TSN could improve are the master-follow configuration, where a single inverter acts as the master for the remaining inverter [36, 44]. The master, acting as voltage source inverter, generates voltage and current reference signals for current source inverters in the microgrid. Traditionally, this configuration is not desirable due to high communication requirements [36]. In this topology, active and reactive power flow are controlled by the MGCC [44].

2.2.3 Real-time State Estimation and Control

Increases in the proliferation of distributed resources make the need for highly accurate state estimation of microgrid systems even more important than today. Traditionally, transmitting the large quantity of grid data back to the centralized state estimator has been an expensive endeavor, due to challenges with both accurate timing of distributed measurements, as well as the fast transmission of these measurements to a centralized server [45]. There has been a large amount of research done into overcoming communication issues inherent with the need for state estimation, to create distributed systems [46]. With IEEE TSN’s ability to both synchronize timing for measurements as well as quickly move this data through a network, the feasibility of a large distributed state estimation system becomes more practical.

True real-time control requires much higher levels of communication latency/reliability guarantees than most common communication backbones can provide. However, there exist several different interpretations of what constitutes real-time system control depending on the field or application. For example, process automation

Table 2.1: Communication bandwidth, latency, and reliability requirements for various smart grid applications [5].

	Parameters		
Application	Bandwidth (kbps/node)	Latency	Reliability
Application Program Interface	10 - 100	1 s -15 s	99.9%
Demand Response	14 - 100 (kbps/node/device)	500 ms - 3 s	99.99%
Wide Area Situational Awareness	600 - 1500	10 ms - 200 ms	99.999 - 99.9999%
Distribution Energy Resources and Storage	9.6 - 100	10 ms - 15 s	99.99%
Distribution Grid Management	9.6 - 100	100 ms - 2 s	99.99 - 99.999%

applications in manufacturing typically require latency from 10-100 ms, with little significant performance variation within that range [47]. Motion control applications require latency under 1 ms for real-time control [47]. However, for real-time control of power systems or microgrids, metering and sensing measurement messages should experience a latency on the order of 10 milliseconds [33, 34].

Real-time microgrid control could take the form of the architecture outlined in [48], where customer preferences, measurements, and forecasting information is used to set microgrid operation to achieve maximum economic optimization. This architecture takes measurements at fixed time steps, runs an optimization algorithm to determine the ideal settings, and sends setpoints and commands back to the local controllers for customer loads and distributed generation resources. The smaller the time step used, the better the overall optimization, at the expense of higher computation resources [48]. If such a system used a TSN based communication system, communication latency could be virtually eliminated as a bottleneck to microgrid control and state estimation. The limit of microgrid control would then be how fast the optimization algorithm could calculate the ideal state for the microgrid at any moment. This would allow microgrid operation to approach ideal real-time operation. Pairing this communication capability with Advanced Metering Infrastructure (AMI) would allow real-time monitoring and control of the microgrid on a house-by-house basis [34], which becomes more important as residential renewables continue to become adopted. AMI also potentially allows the real-time control of residential/industrial loads to manage energy consumption within the microgrid.

2.2.4 Challenges Facing TSN Adoption

The greatest challenge facing any deployment of an IEEE Time Sensitive Networking communication system for microgrid applications would be cost. TSN, as of this writing, is only available on 802.3 Ethernet based networks, thus necessitating a physical connection between all communication nodes. The high cost of laying physical connections as opposed to using some kind of wireless based communication system would be potentially difficult for a utility or operator to justify. However, due to TSN’s backwards compatibility with existing LAN networks, any existing Ethernet based deployment could upgrade to a TSN network while using existing infrastructure. It is also feasible that faster response times and more accurate levels of real-time control enabled by the technology would justify the price of a TSN based communication network.

In addition to cost, the current state of the technology is immature, at least in the commercial space. This of course will change in the future, as more manufacturers develop and release TSN enabled devices, and support and familiarity with the capabilities and system administration of TSN networks becomes more common. In some sense the current capabilities of TSN enabled device, while functional, are more appropriate for lab or research related work, rather than power system deployment where stability and reliability are paramount. As the commercial availability of TSN hardware grows, and the comprehensiveness of TSN system administration software matures, the technology will become more viable as a true microgrid and power system communication option.

2.3 Chapter Summary

Chapter 2 presented the technical background to the TSN 802.1AS and 802.1Qbv standards, as well as a brief overview of the remaining TSN standards not heavily covered in this project. A basic understanding of the methods by which TSN time synchronization and time aware traffic shaping occur is necessary to understand the potential and limits of the technology for various applications.

In addition, the potential use case of this technology in smart grid or microgrid power system applications was discussed. The greatest potential for TSN technology in these applications is rapid dissemination of state information from distributed monitoring devices to a central controller, and rapid control speed throughout a microgrid from central control to distributed entity. In addition, time synchronization between all nodes enables unique features, such as highly coordinated actions to be taken throughout a microgrid at the same moment, as well as inverter synchronization during islanding.

3 Materials and Design Methodology

3.1 Introduction to Topology and Test Plan

This section lays out the basic testing plan to validate and demonstrate the capabilities of TSN and its potential for integration into microgrid power systems. The tests conducted will demonstrate TSN's advantages compared to the standard link layer Ethernet capabilities of TCP and UDP transport layer protocols, in the context of control speed and synchronization. The first tests to be conducted will be a comparison of the latency guarantees capable with TSN. To this end, the maximum latencies of TSN, TCP, and UDP will be compared under a variety of payload sizes and network congestion levels. This is to demonstrate that TSN can maintain latency requirements well under a single standard (60 Hz) cycle, even at high levels of network use. The second set of tests to be conducted demonstrate the timing abilities of TSN, both in their precision under different network congestions, as well as their ability to effectively generate high precision waveforms for use with synchronizing multiple grid-forming inverters across a large area. Several tests will be conducted, validated with an oscilloscope. These include tests to toggle a square wave every second, shown to be synchronized with a high degree of precision and demonstrating two synchronized sine waveforms synchronized to within 1 degree of phase shift. In addition, the ability to utilize the deterministic latency guarantees to have a remote station track an arbitrary waveform setpoint in real time across a network while remaining in sync is shown.

3.2 Introduction to Equipment and Materials

This section introduces the equipment used in support of this project. It should be noted that as TSN is a relatively new development of the Ethernet standard, current availability of TSN capable equipment is limited.

3.2.1 National Instruments Compact RIO 9035-Sync

To run the code and emulate the controllers necessary for microgrid control, two National Instruments (NI) Compact RIO 9035-Sync FPGA (cRIO) controllers were used. These devices are programmed using the NI LabVIEW development environment, and are equipped with the special network interface required to implement the TSN hardware timestamp and additional TSN features. These controllers are each additionally fitted with a single NI 9263 analog output module, capable of outputting 4 channels of analog data with +/-10V 16 bit resolution at 100kS/s. These modules were used to physically measure the timing signals and synchronization signals via oscilloscope. In addition, several additional software modules are required to use the features outlined in this implementation, including the FGPA, Real-time, and TSN modules.

3.2.2 Cisco Industrial Switch 4000

The time aware bridges used in this implementation are Cisco Industrial Switch 4000 switches (IE-4000-8GT4G-E). These switches are each equipped with 12 gigabit Ethernet ports enabled with the hardware required to support TSN features. It should be noted that these switches only support three of the TSN standards discussed in Chapter 2: IEEE 802.1AS, IEEE 802.1Qbv, and IEEE 802.1Qcc. While this is not the full suite of TSN standards, it does have the base standards required to be called a "TSN implementation" [20]. Four of the ports on the switches are also dual ports, which can be used with Small Form-factor Pluggable (SFP) transceivers instead of copper RJ-45 connectors. These transceivers allow for the support of fiber connections between switches, greatly increasing the maximum range of a physical link between two switches from the copper Ethernet maximum of 300 m to tens of kilometers.

3.2.3 Cisco Central Network Controller

The Cisco Central Network Controller (CNC), (sometimes referred to in TSN literature as Central Network Configurator) is a service used for the network discovery and configuration of TSN network streams. The CNC runs inside an Ubuntu Linux virtual machine on a host workstation running Oracle Virtualbox. The CNC allows for scheduling of the 802.1Qbv flows, and specifying of latency requirements and additional options. The CNC falls under the 802.1Qcc standard, implementing the system administration features of creating and defining TSN flows from a centralized portal. The software as used in this implementation is in beta release, and was not in final release form. Hence, there were some limitations on the full functionality of the CNC.

3.2.4 Network Congestion Generator

A Dell R610 sever was used to aid in the consistent and repeatable loading of the TSN network for congestion testing. VMWare ESXi 6.5 was installed on this server as a bare-metal hypervisor, allowing for the creation of virtual machines. Four Ubuntu Server virtual machines were created, each installed with the iPerf network bandwidth measurement tool. Two virtual machines were assigned to port eth1 of the server, while the other two were assigned to port eth2 of the server. Eth0 was reserved for direct connection to the web based management interface (webUI), in the event that high levels of network congestion could inhibit the ability to access and shut down the congestion tests.

3.3 PRELIMINARY SETUP

This section outlines the basic steps taken in preparation for tests characterizing the performance of TSN time synchronization and latency guarantees. This includes the physical setup of the testing topology, as well as the LabVIEW code used for the testing. In addition, setup of the TSN flows within the Cisco CNC and generation of network congestion traffic are discussed.

3.3.1 Physical Topology

The physical setup of the testing is shown in Figure 3.1. In the test setup, one cRIO 9035-sync is connected (via its single TSN enabled Ethernet port) to a port on a single IE4000. The two IE4000 switches are then connected together via a trunk port. In

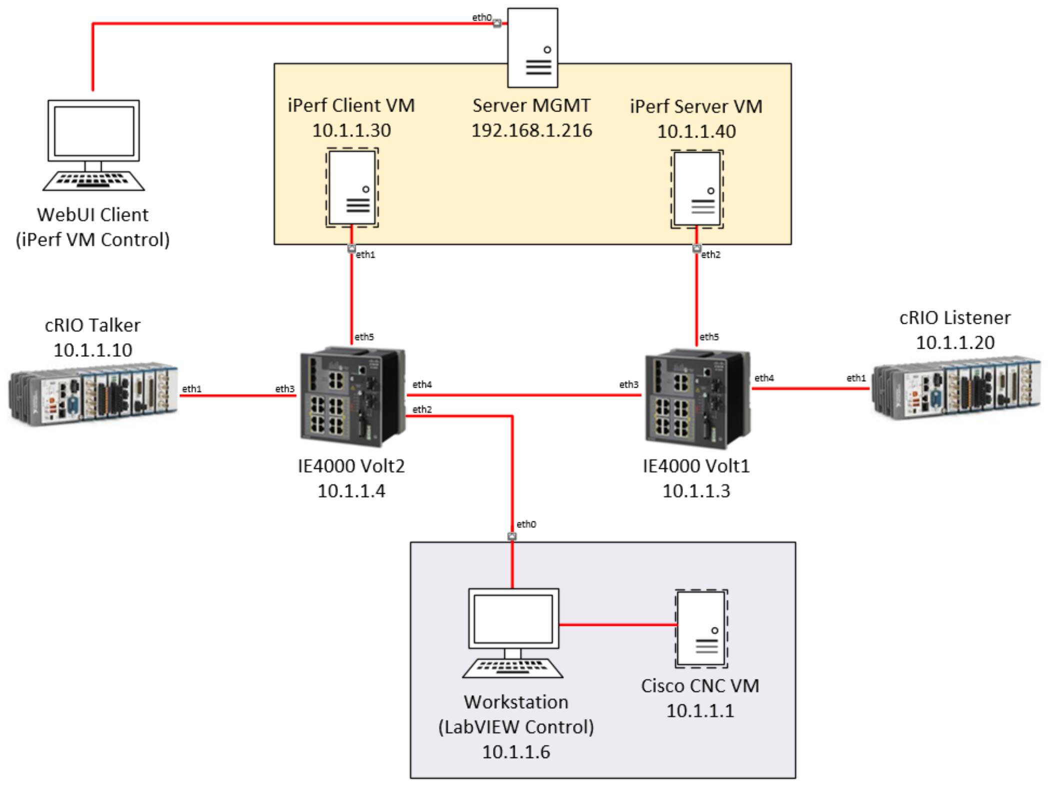


Figure 3.1: Physical setup of the TSN test network.

addition, two Ethernet links (one per IE4000) leave the switches to terminate at ports eth1 and eth2 on the Dell server. These links will facilitate the congestion needed for testing (Discussed further in 3.3.2).

The workstation PC (used for running the LabVIEW software used to interface with cRIOs and the Cisco CNC software used to interface with the switches) is connected via a USB to Ethernet adapter to one of the IE4000 switches. The workstation is used purely to start and stop the tests that run on the cRIO, and configure the hardware.

The Dell server is connected to a second PC using a USB to Ethernet adapter connected to port eth0 on the server. This is used to access the ESXi webUI, used to configure and control the Ubuntu virtual machines running the iPerf network measurement tool. This management interface is not on the same network as the TSN

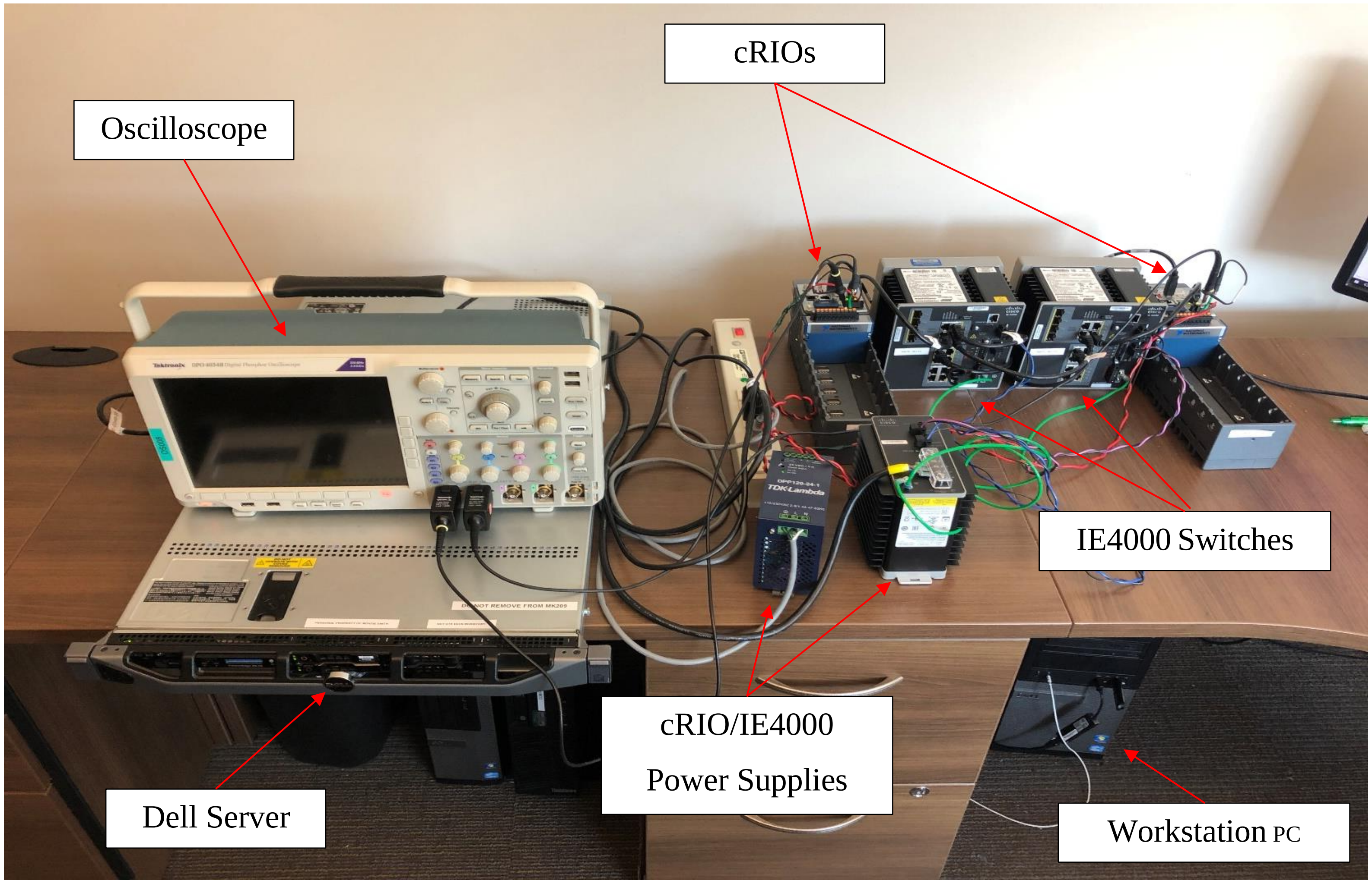


Figure 3.2: Physical testing setup.

devices under test, to avoid the high levels of network congestion somehow preventing access to the machine to stop the generated network streams.

Each cRIO 9035-sync is installed with a single National Instruments 9263 Analog output module. This module is used to generate analog signals to interface with measurement and real time simulation platforms that do not support TSN, such as oscilloscopes and the RTDS cabinet. Both cRIOs are powered from a single 24V TDK-Lambda DPP120-24-1 DC power supply. Both IE4000s are powered from a single Cisco 54V DC power supply.

3.3.2 Network Congestion Generation

For the testing of the relative improvements of TSN compared with other more common communication protocols, a method of generating network impairment was needed. It was determined that fully saturating a single switch would be impractical (if not impossible), as the IE4000 switches being used are designed to provide full line speed forwarding capability on all 12 ports at once. Instead of trying to inhibit the entire switch, it was determined that it would be easier to attempt to congest the trunk link between two IE4000 switches, which would be necessarily limited to the line speed of a single

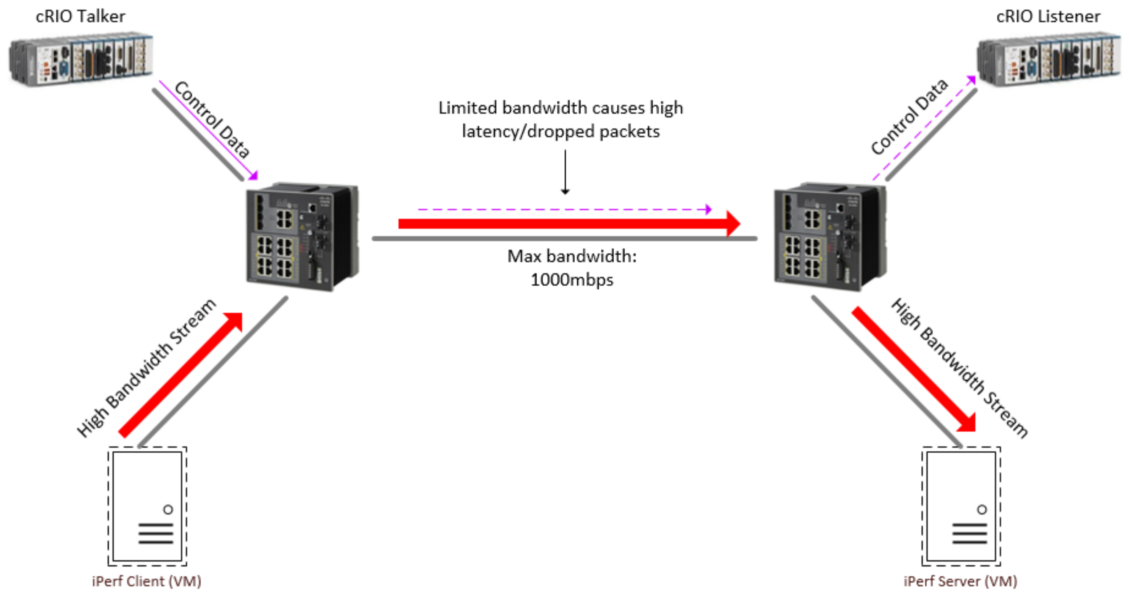
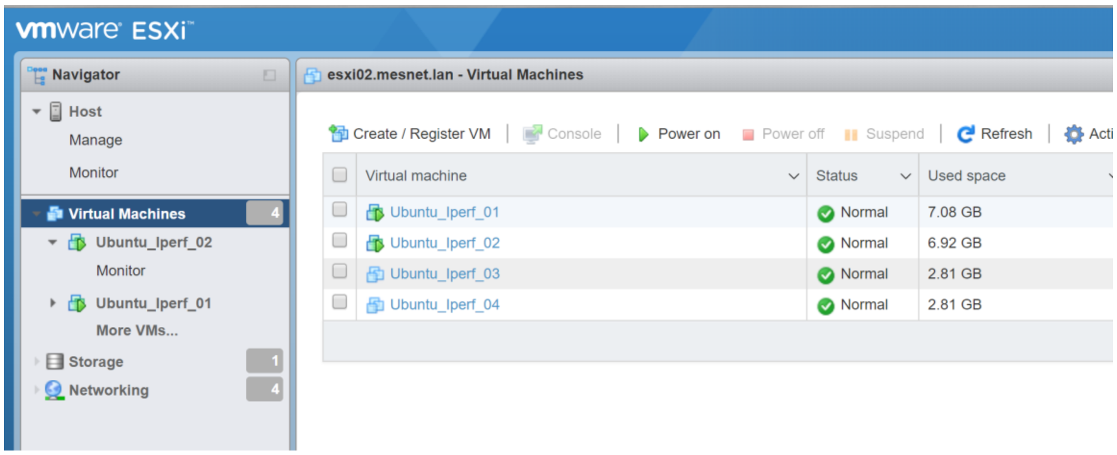


Figure 3.3: Generating network congestion along the switch trunk port.



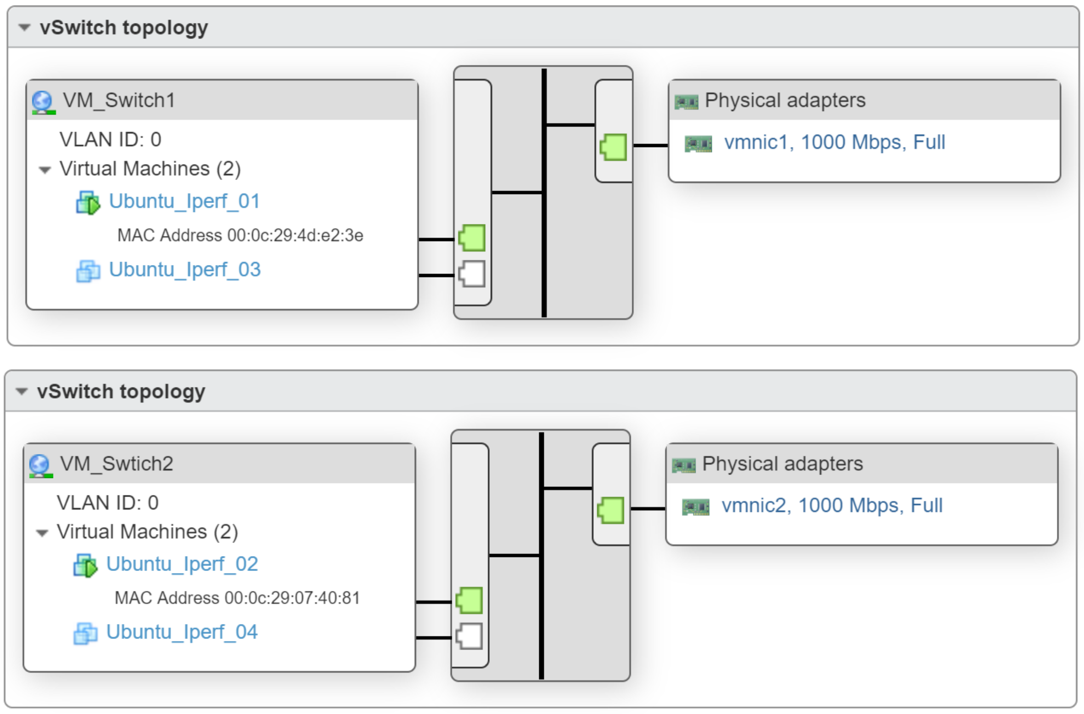
Virtual machine	Status	Used space
Ubuntu_Iperf_01	Normal	7.08 GB
Ubuntu_Iperf_02	Normal	6.92 GB
Ubuntu_Iperf_03	Normal	2.81 GB
Ubuntu_Iperf_04	Normal	2.81 GB

Figure 3.4: Four Ubuntu virtual machines running on one server. Each runs the iPerf utility.

Ethernet link (1000 mbps). Figure 3.3 shows the eventual setup. By attempting to pass up to and more than 1000mbps of data into the trunk, some packets would either be queued or lost, inducing latency.

To facilitate this, ESXi 6.5 (a hypervisor software used to create virtual machines) was installed as the base OS on a Dell R610 server used for testing. (Note that this machine was chosen due to its availability and installed quad-port NIC, however it is far more powerful than required to actually generate the necessary amount of traffic). Within the ESXi OS, several virtual machines (VMs) were created, each running Ubuntu Server 16.04 LTS, a readily available command line Linux distribution. Figure 3.4 shows the VMs within the ESXi webUI. Though four VMs were created, only two were eventually used for testing. Each of these VMs was virtually connected to one of two physical Ethernet ports on the server: two to eth1 and two to eth2. In this way, data from the virtual machines could be sent out to the physical TSN test network. In addition, VMs assigned to different physical network ports could not reach each other directly through the server, instead having to exit into the TSN network before returning back to the other physical port. Figure 3.5 shows the configuration setup within the ESXi webUI, showing two VMs assigned to each physical network port.

Within each of these Linux virtual machines, the software utility iPerf was installed (via documentation available on the iPerf website). iPerf is a command line


vSwitch topology

VM_Switch1

VLAN ID: 0

Virtual Machines (2)

Ubuntu_Iperf_01

MAC Address 00:0c:29:4d:e2:3e

Ubuntu_Iperf_03

Physical adapters

vmnic1, 1000 Mbps, Full

vSwitch topology

VM_Swtich2

VLAN ID: 0

Virtual Machines (2)

Ubuntu_Iperf_02

MAC Address 00:0c:29:07:40:81

Ubuntu_Iperf_04

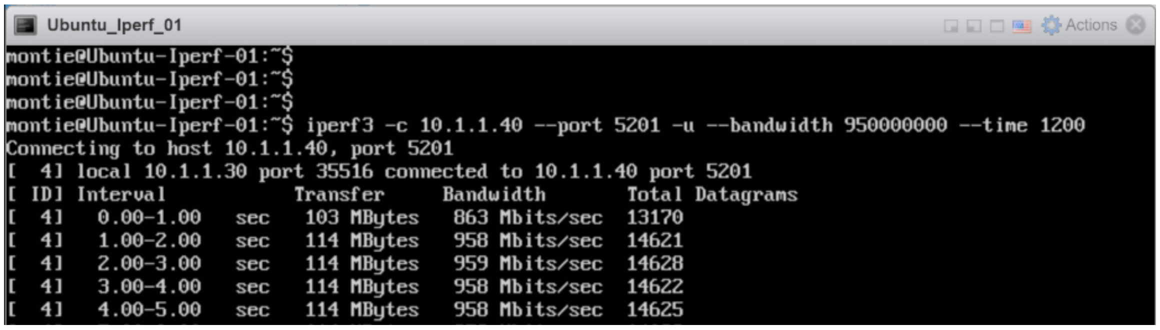
Physical adapters

vmnic2, 1000 Mbps, Full

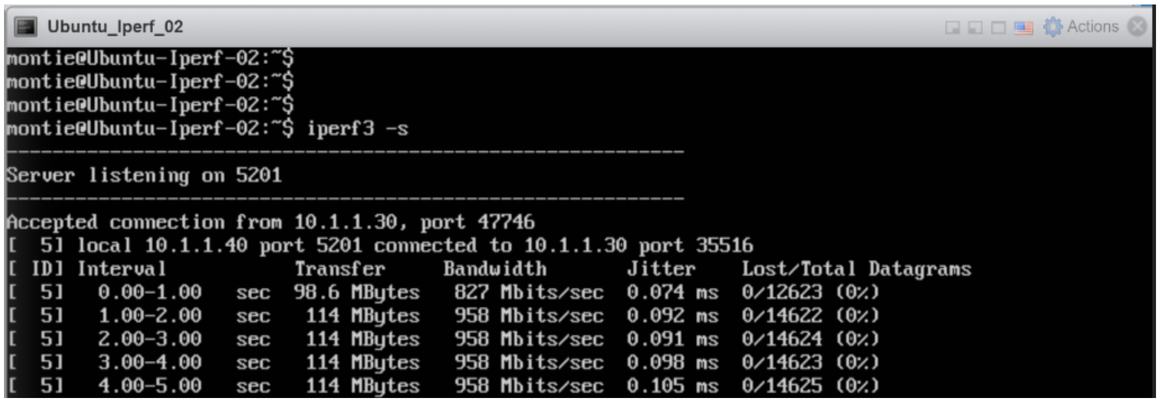
Figure 3.5: Two VMs each are virtually connected to eth1 (vmnic1) and eth2 (vmnic2), the physical Ethernet ports on the rear of the server.

bandwidth measurement utility, used to test the maximum throughput of a link using various parameters and protocols set by the user. iPerf must be installed on machines on both sides of the link being tested, one acting as a (sending) client and the other acting as a (receiving) server. As previously discussed, the client virtual machine is virtually connected (within ESXi) to one of the two physical Ethernet ports on the rear of the Dell R610 server. This port is then connected to the first IE4000 switch, which is then connected via a trunk to the second IE4000 switch. The second switch is then connected to a second port on the Dell server, which is virtually connected to the second iPerf installation virtual machine. In this way, traffic generated from the iPerf client sent to the iPerf server (all installed on a single machine) must traverse the physical link between the two switches. Figure 3.1 shows these various connections, including to the virtual machines running a single computer.

Using various command line flags when initializing iPerf, the stream can be controlled to inhibit the trunk link to varying degrees. While normally iPerf is used to measure bandwidth in a short burst, it can be configured to generate a specific level of



```
montie@Ubuntu-Iperf-01:~$
montie@Ubuntu-Iperf-01:~$
montie@Ubuntu-Iperf-01:~$
montie@Ubuntu-Iperf-01:~$ iperf3 -c 10.1.1.40 --port 5201 -u --bandwidth 950000000 --time 1200
Connecting to host 10.1.1.40, port 5201
[  4] local 10.1.1.30 port 35516 connected to 10.1.1.40 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   103 MBytes   863 Mbits/sec  13170
[  4]   1.00-2.00   sec   114 MBytes   958 Mbits/sec  14621
[  4]   2.00-3.00   sec   114 MBytes   959 Mbits/sec  14628
[  4]   3.00-4.00   sec   114 MBytes   958 Mbits/sec  14622
[  4]   4.00-5.00   sec   114 MBytes   958 Mbits/sec  14625
```



```
montie@Ubuntu-Iperf-02:~$
montie@Ubuntu-Iperf-02:~$
montie@Ubuntu-Iperf-02:~$
montie@Ubuntu-Iperf-02:~$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 10.1.1.30, port 47746
[  5] local 10.1.1.40 port 5201 connected to 10.1.1.30 port 35516
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec  98.6 MBytes   827 Mbits/sec  0.074 ms  0/12623 (0%)
[  5]   1.00-2.00   sec   114 MBytes   958 Mbits/sec  0.092 ms  0/14622 (0%)
[  5]   2.00-3.00   sec   114 MBytes   958 Mbits/sec  0.091 ms  0/14624 (0%)
[  5]   3.00-4.00   sec   114 MBytes   958 Mbits/sec  0.098 ms  0/14623 (0%)
[  5]   4.00-5.00   sec   114 MBytes   958 Mbits/sec  0.105 ms  0/14625 (0%)
```

Figure 3.6: iPerf Client (top) and Server (bottom). Each image shows iPerf being started/running.

bandwidth for extended periods of time. The specified line speed to be sent, with a theoretical max of 1000 mbps, is proportional to the percent of the total link that is congested. For example, a 500 mbps stream translates to 50% link congestion for a 1000mbps theoretical maximum bandwidth. However in this implementation, the max line speed achievable over the trunk link was found to be 958 mbps, meaning the percentage congestion corresponding to “max congestion” is approximately 95%, and is reflected in the results in Chapter 4. **Error! Reference source not found.** shows iPerf b eing configured and started via the Ubuntu command line interface (CLI). In addition, the **Error! Reference source not found.** shows the display output while the service is running. While running, a summary of the link statistics is displayed every second. It can be seen that 958 mbps was the maximum achievable bandwidth, even when 1000 mbps was input as the desired bandwidth to stream. The protocol used to generate the stream is UDP.

3.3.3 LabVIEW Code

This section will describe the LabVIEW program files (known as "Virtual Instruments" or VIs) created to enable deterministic 802.1Qbv TSN flows and generate synchronization signals based on 802.1AS common time references. In addition, the portions of the VIs used to accurately measure latency will be described in detail.

For the tests conducted, one cRIO acts as the "sending" device and the other acts as the "receiving" device. The first cRIO acts as a TSN "talker" and the second acts as a TSN "listener". There are three VIs that are being used on each cRIO for each test: the main (master/follower control) VI, the FPGA VI, and the communication VI. The top level main VI is used for user input and required background tasks. Figure 3.7 shows the VI functions of both cRIOs, as well as their relation to each other.

On the talker cRIO, the user inputs are collected into a cluster, which is then fed into the communication VI. There are three communication VIs, one for each of the

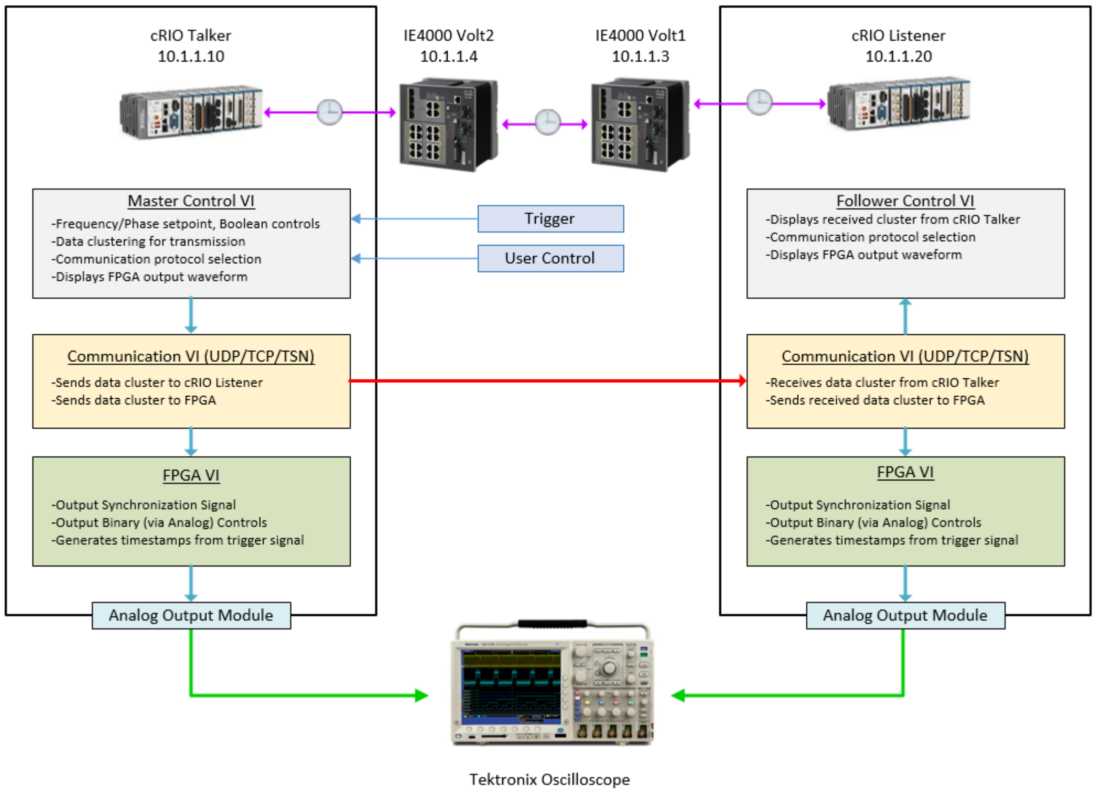

cRIO Talker
10.1.1.10
IE4000 Volt2
10.1.1.4
IE4000 Volt1
10.1.1.3
cRIO Listener
10.1.1.20
Master Control VI
-Frequency/Phase setpoint, Boolean controls
-Data clustering for transmission
-Communication protocol selection
-Displays FPGA output waveform
Trigger
User Control
Follower Control VI
-Displays received cluster from cRIO Talker
-Communication protocol selection
-Displays FPGA output waveform
Communication VI (UDP/TCP/TSN)
-Sends data cluster to cRIO Listener
-Sends data cluster to FPGA
Communication VI (UDP/TCP/TSN)
-Receives data cluster from cRIO Talker
-Sends received data cluster to FPGA
FPGA VI
-Output Synchronization Signal
-Output Binary (via Analog) Controls
-Generates timestamps from trigger signal
FPGA VI
-Output Synchronization Signal
-Output Binary (via Analog) Controls
-Generates timestamps from trigger signal
Analog Output Module
Analog Output Module
Tektronix Oscilloscope

Figure 3.7: VI topology of the cRIOs.

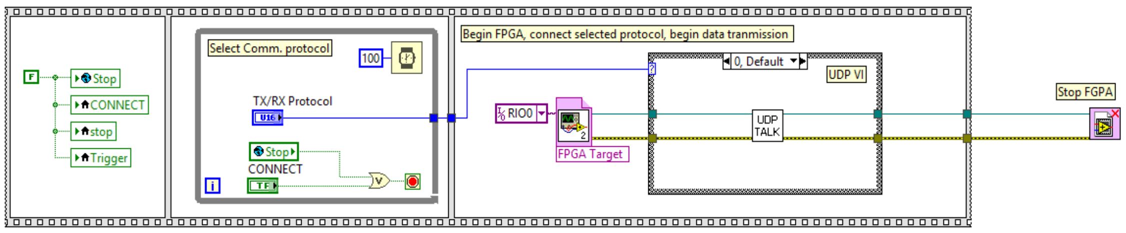


Figure 3.8: LabVIEW code for selecting communication protocol to use on both the talker and listener cRIOs. This code is a part of the main VI shown in Figure 3.9.

following communication protocols: UDP, TCP, and TSN. Only one may be running at a time. The user selects which protocol is to be used on each cRIO before establishing the connection. The cluster of data sent to the listening cRIO is handled the same, regardless of which communication protocol was chosen to actually move the data. In the same VI that facilitates the chosen communication protocol, the data cluster variables are input into the FPGA VI. The FPGA handles timing measurement functions, saving timestamps used for latency measurement. It also generates the waveforms sent to the NI analog output module.

At the second listening cRIO, the cluster of data is received in the communication VI (via the chosen protocol) and input again into the FPGA VI of the listener cRIO. The configuration of the code is such that the data cluster being sent to the FPGA VI occurs during the same loop as the transmission of the data out of the first cRIO, and during the same loop upon reception at the second cRIO.

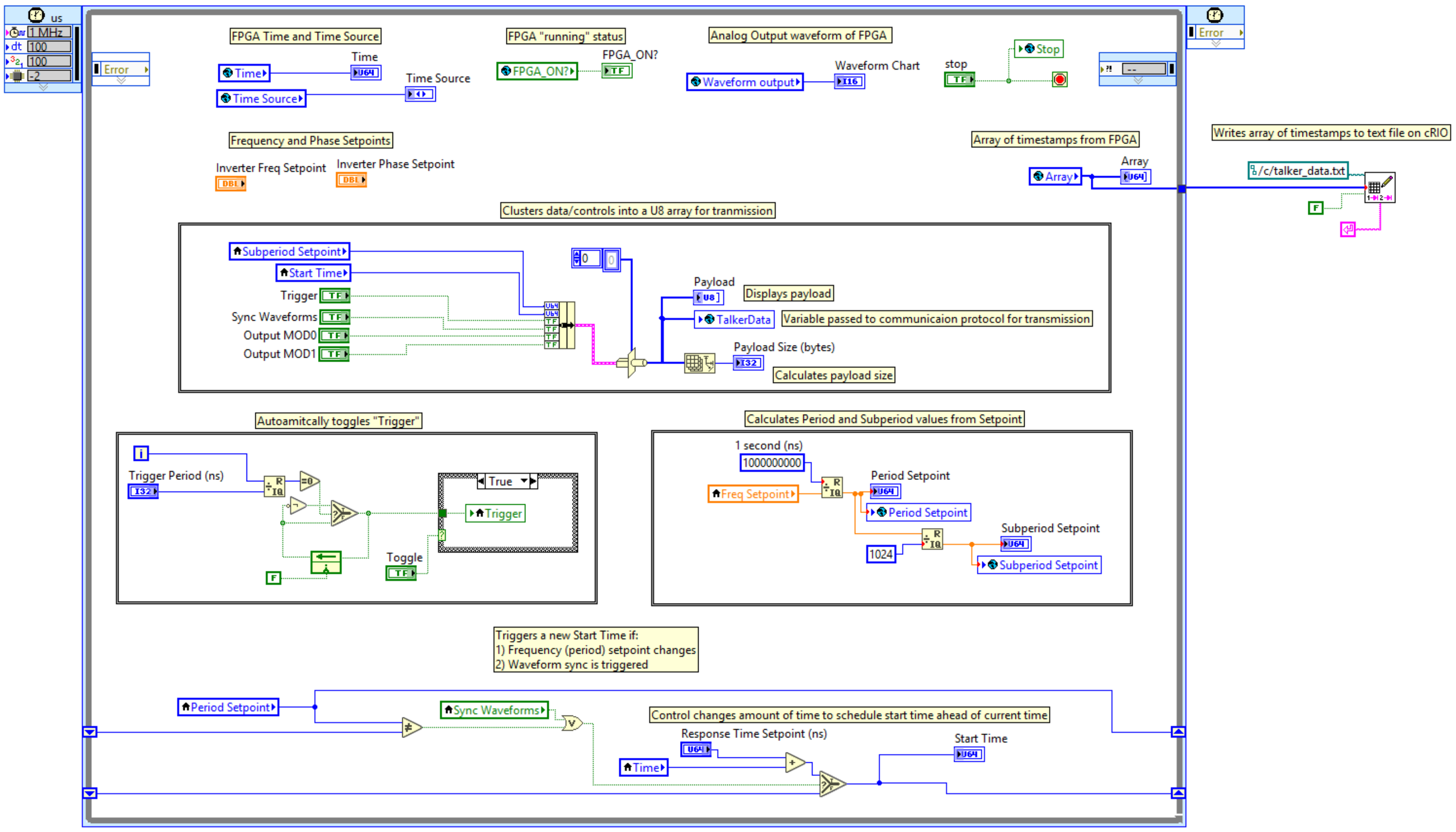
The FPGA VI monitors one Boolean variable (*Trigger*) for a low to high transition. When this variable transitions from low to high, the exact time of this change is recorded in an array. When this change of variable is received at the second cRIO, the second timestamp is recorded. The difference in these two timestamp values should equal the communication latency of the protocol being tested. Note that due to the loop iteration delay caused by the follower cRIO, the latency measurement has an additional 100 µs added, which is consistent between all communication methods.

Main (Master/Follower) VI

Figure 3.9 shows the main (master) control VI LabVIEW code. This code receives input from the user via the LabVIEW front panel, such as commands for toggle

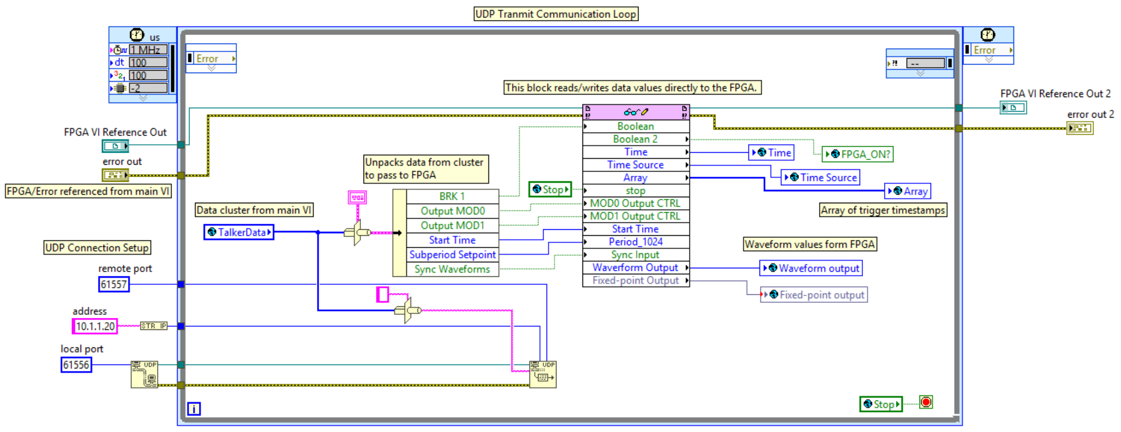
switches and frequency/phase setpoints. In addition, this VI performs additional services such as converting the user input data into a cluster for transmission to the listener cRIO, as well as starting the user selected communication protocol for data transmission. Additional functions are present, such as the ability to toggle the *Trigger*
value at a set interval, or alter the frequency randomly within a small range (simulating small grid frequency fluctuations). Figure 3.8 shows a small subsection of the main VI, where the communication protocol is selected. After the protocol is selected and the user triggers the *Connect* command, the FPGA code file is initialized, and references to this file are passed to the communication VI of the protocol selected. Figure 3.12 shows the front panel user interface for the main control VI.

The main VI of the listener cRIO has fewer functions than on the talker cRIO. It is used to select which communication protocol is used to listen on (using the same code as shown in Figure 3.8, but using listener specific communication subVIs). In addition, the follower main VI displays the received data visually to the user on the front panel. The follower VI has no further functions.



FPGA Time and Time Source
Time
Time Source
FPGA "running" status
FPGA_ON?
Analog Output waveform of FPGA
Waveform output
Waveform Chart
stop
Stop
Error
Frequency and Phase Setpoints
Inverter Freq Setpoint
Inverter Phase Setpoint
Array of timestamps from FPGA
Array
Writes array of timestamps to text file on cRIO
/c/talker_data.txt
Clusters data/controls into a U8 array for tranmission
Subperiod Setpoint
Start Time
Trigger
Sync Waveforms
Output MOD0
Output MOD1
Payload
Displays payload
TalkerData
Variable passed to communicaion protocol for transmission
Payload Size (bytes)
Calculates payload size
Autoamitcally toggles "Trigger"
Trigger Period (ns)
True
Toggle
Calculates Period and Subperiod values from Setpoint
1 second (ns)
1000000000
Freq Setpoint
Period Setpoint
Subperiod Setpoint
1024
Triggers a new Start Time if:
1) Frequency (period) setpoint changes
2) Waveform sync is triggered
Period Setpoint
Sync Waveforms
Control changes amount of time to schedule start time ahead of current time
Response Time Setpoint (ns)
Start Time
1 MHz
dt
100
-2

Figure 3.9: LabVIEW code that implements top level main functions on talker cRIO.



Fiaure 3.10: UDP Communication VI for Talker cRIO.

<u>*Communication VI*</u>

There are three versions of the communication VI: one each for UDP, TCP, and TSN communication. These are implemented differently in LabVIEW, but all function roughly the same. The FPGA file is initiated, and references to it are passed into the VI for the chosen communication protocol (see Figure 3.8). This reference is used to send and receive values to the cRIO FPGA using a *Read/Write Control Function* block. This allows for values to be written and read directly from the FPGA in the same iterative loops as the communication functions are run. In doing so, the control signals (bundled previously in the top level main VI) are both transmitted to the listener cRIO and passed to the talker FPGA in the same loop. This is necessary later when testing latency and timing functionality of TSN. The FPGA is used for all timing relayed measurements and generation of synchronized sine waveforms.

As can be seen in Figure 3.10, Figure 3.11 and Figure 3.13, the data variable *TalkerData* is received from the main VI, and is then split back into its constituent variables to be passed into the FPGA. The whole cluster is also (in the UDP/TCP cases) converted to a string to be transmitted. In the TSN case, the unsigned byte array is directly input into the TSN function blocks for transmission to the listener cRIO.

The listener cRIO has three similar communication VIs for each protocol. Rather than inputting the data into the TCP/UDP/TSN function blocks, the data is output from the protocol function blocks. Writing these values to the FPGA VI is handled identically to the talker communication VIs

FPGA VI

The FPGA VI, shown in Figure 3.14, controls the timing aspects of this TSN implementation. In the FGPA VI, the frequency setpoints set in the main VI (on the talker cRIO) or received via the communication VI (on the listener cRIO) are interpreted in a way to generate synchronized waveform outputs. In addition, this VI contains code to save 802.1AS synchronized timestamps when the *Trigger* signal transitions from 0 to 1. In this way, synchronized timestamps from the talker and listener cRIOs can be compared to determine latency.

The FGPA VI generates synchronized waveforms by using a sinewave look up table (LUT) with 10-bit precision (1024 states). The frequency setpoint from the top level main VI is converted into a period, and this period is divided by 1024 to determine a subperiod. The subperiod is the length of time that any given state of the LUT needs to be output until the LUT index needs to be iterated. Using the synchronized timing present in 802.1AS, the output of distributed cRIOs can be configured to begin outputting a waveform of a given period (frequency) at the same moment. Due to the synchronized concept of time, each cRIO will iterate to the next LUT index at the same moment, and so on, outputting sinewaves with synchronized frequency and phase.

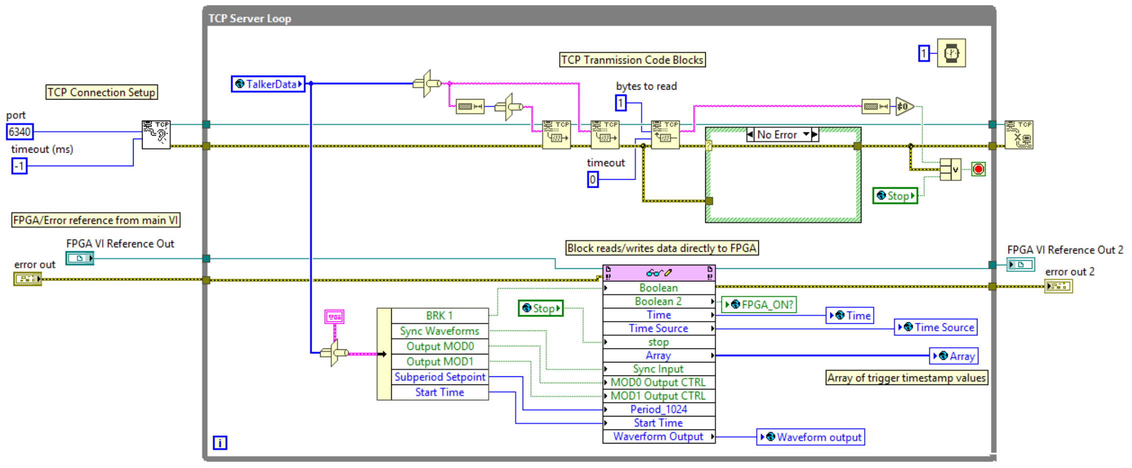
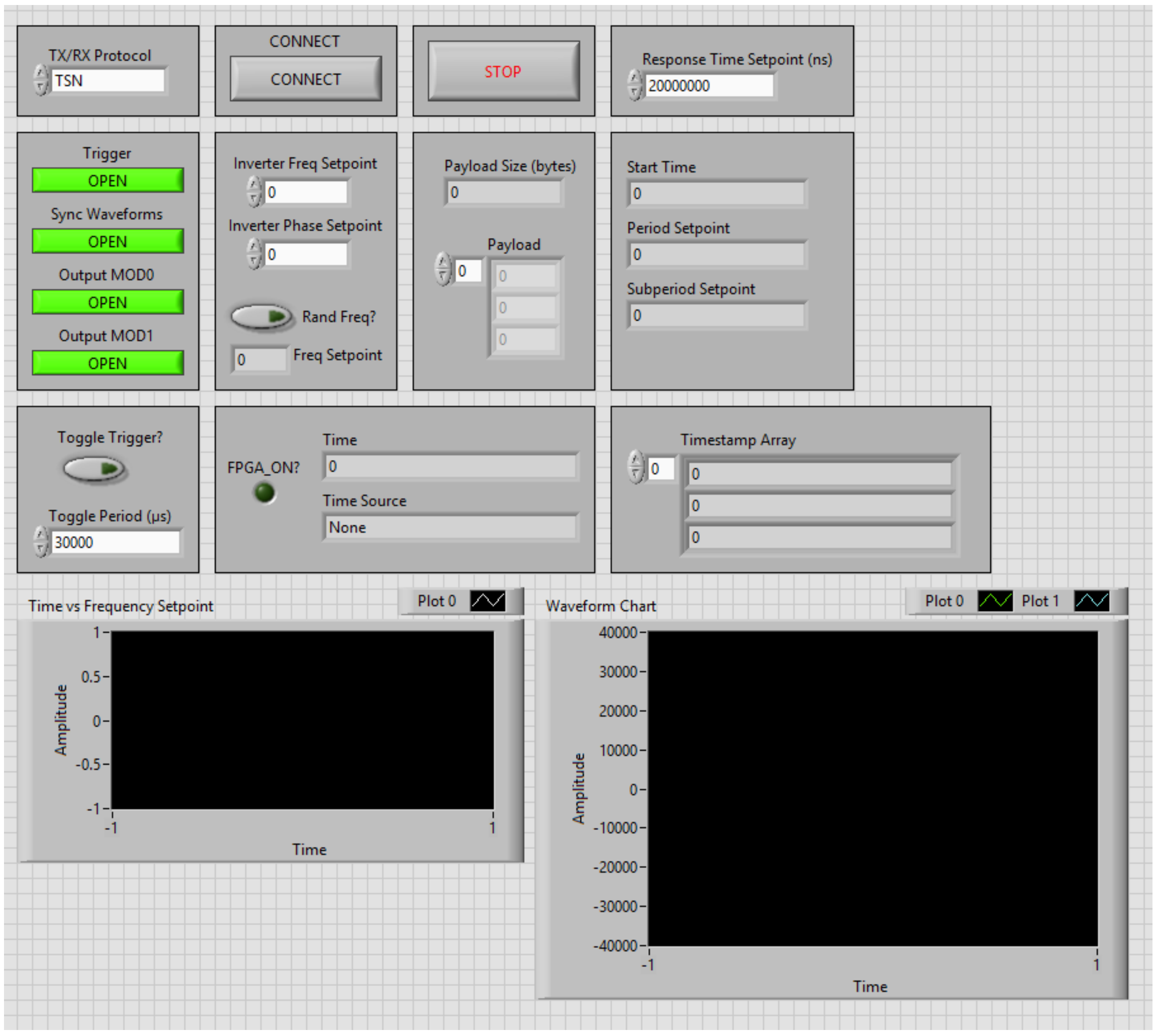


Figure 3.11: TCP Communication VI for Talker cRIO.


TX/RX Protocol: TSN

CONNECT: CONNECT

STOP

Response Time Setpoint (ns): 20000000

Trigger: OPEN

Sync Waveforms: OPEN

Output MOD0: OPEN

Output MOD1: OPEN

Inverter Freq Setpoint: 0

Inverter Phase Setpoint: 0

Rand Freq?

Freq Setpoint: 0

Payload Size (bytes): 0

Payload: 0 | 0 | 0 | 0

Start Time: 0

Period Setpoint: 0

Subperiod Setpoint: 0

Toggle Trigger?

Toggle Period (µs): 30000

FPGA_ON?

Time: 0

Time Source: None

Timestamp Array: 0 | 0 | 0 | 0

Figure 3.12: LabVIEW front panel for talker cRIO main VI.

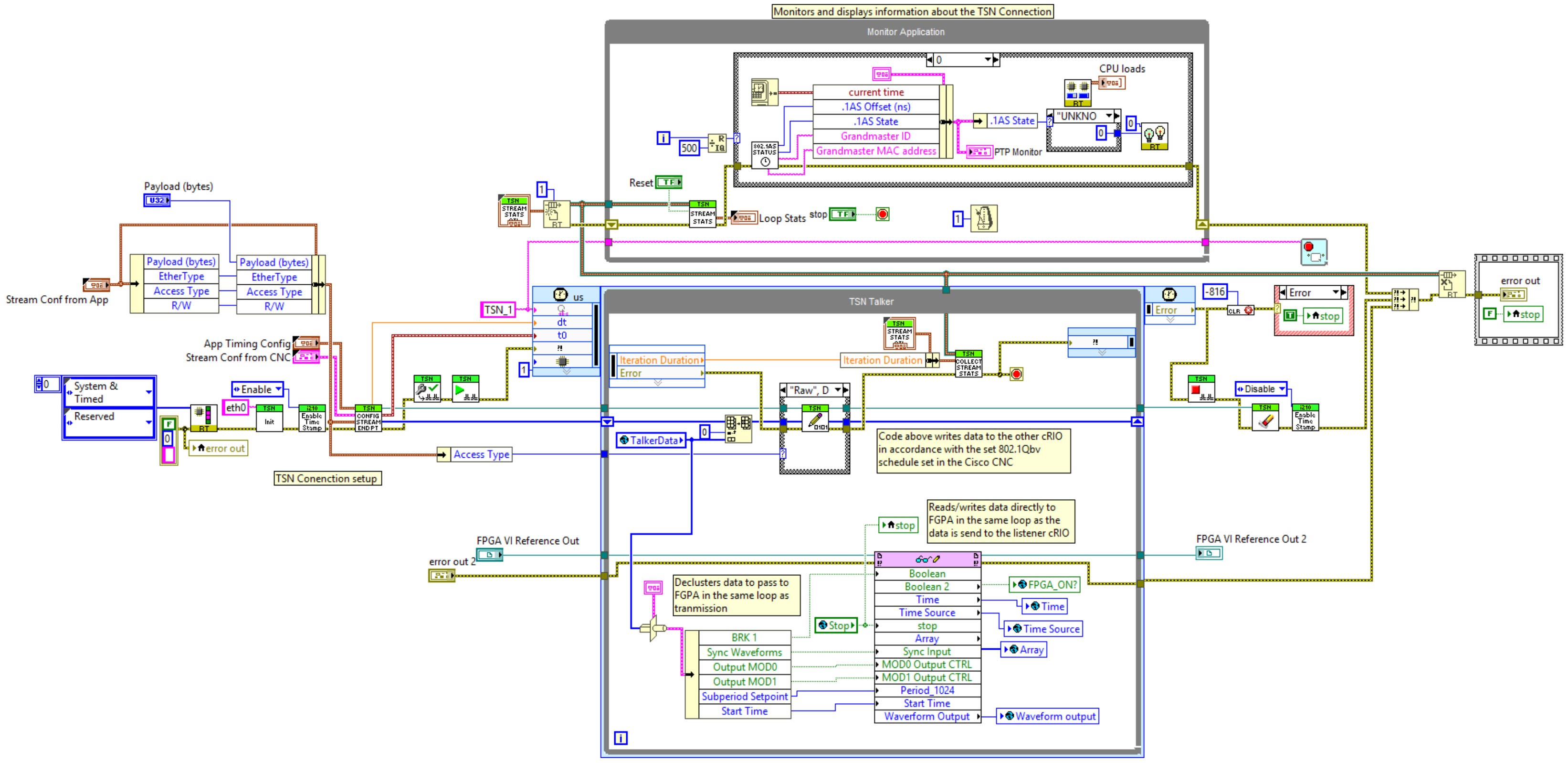
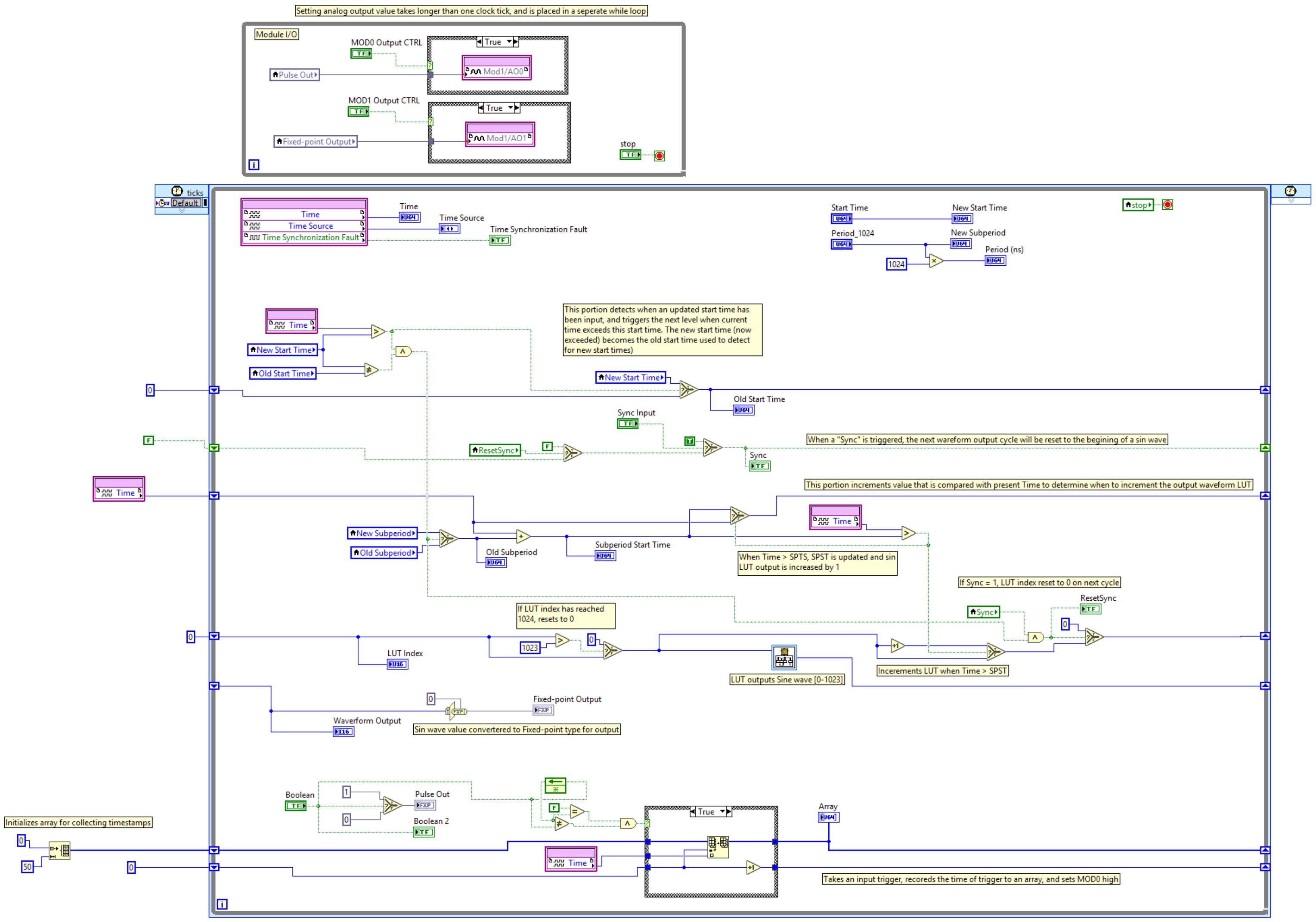


Figure 3.13: TSN Communication VI for Talker cRIO. Much of the basis for this VI is taken from examples on the NI TSN support forums.



Setting analog output value takes longer than one clock tick, and is placed in a seperate while loop

This portion detects when an updated start time has been input, and triggers the next level when current time exceeds this start time. The new start time (now exceeded) becomes the old start time used to detect for new start times)

When a "Sync" is triggered, the next waveform output cycle will be reset to the begining of a sin wave

This portion increments value that is compared with present Time to determine when to increment the output waveform LUT

When Time > SPTS, SPST is updated and sin LUT output is increased by 1

If Sync = 1, LUT index reset to 0 on next cycle

If LUT index has reached 1024, resets to 0

LUT outputs Sine wave [0-1023]

Increments LUT when Time > SPST

Sin wave value convertered to Fixed-point type for output

Initializes array for collecting timestamps

Takes an input trigger, recoreds the time of trigger to an array, and sets MOD0 high

Figure 3.14: FPGA LabVIEW code, used for saving latency measurement timestamps and generating synchronized sine waves.

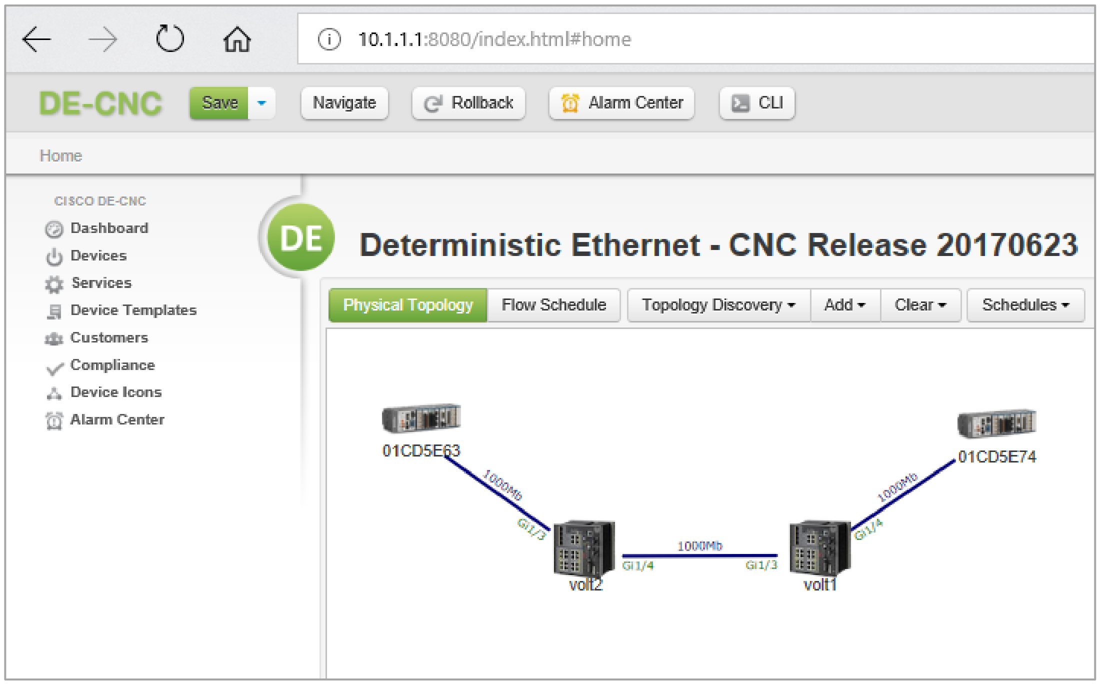


Figure 3.15: CNC webUI, showing the discovered TSN network.

3.3.4 Central Network Controller

The Cisco Central Network Controller (CNC) is the central user interface created by Cisco to conform to the IEEE 802.1Qcc portion of the TSN standards, specifying a single management interface to administer a TSN network. The CNC is run in an Ubuntu 14.04 LTS virtual machine on the workstation PC, using Oracle Virtualbox. This was downloaded as an OVA (open virtualization appliance) template from NI's TSN support forums.

The CNC allows for the discovery of the TSN network, using the link layer discover protocol (LLDP). Upon running the network discovery tool in the CNC, the IE4000 switches and cRIO 9035-sync devices, as well as the physical links between them, are discovered and graphically presented, as shown in Figure 3.15. It is here that deterministic TSN flows may be specified, as shown in Figure 3.16. Note that TSN flows are unidirectional.

Figure 3.17 shows the flow configuration menu in the CNC. The flow is set up specifying the talker cRIO, and the listener cRIO (one or multiple). In addition, the

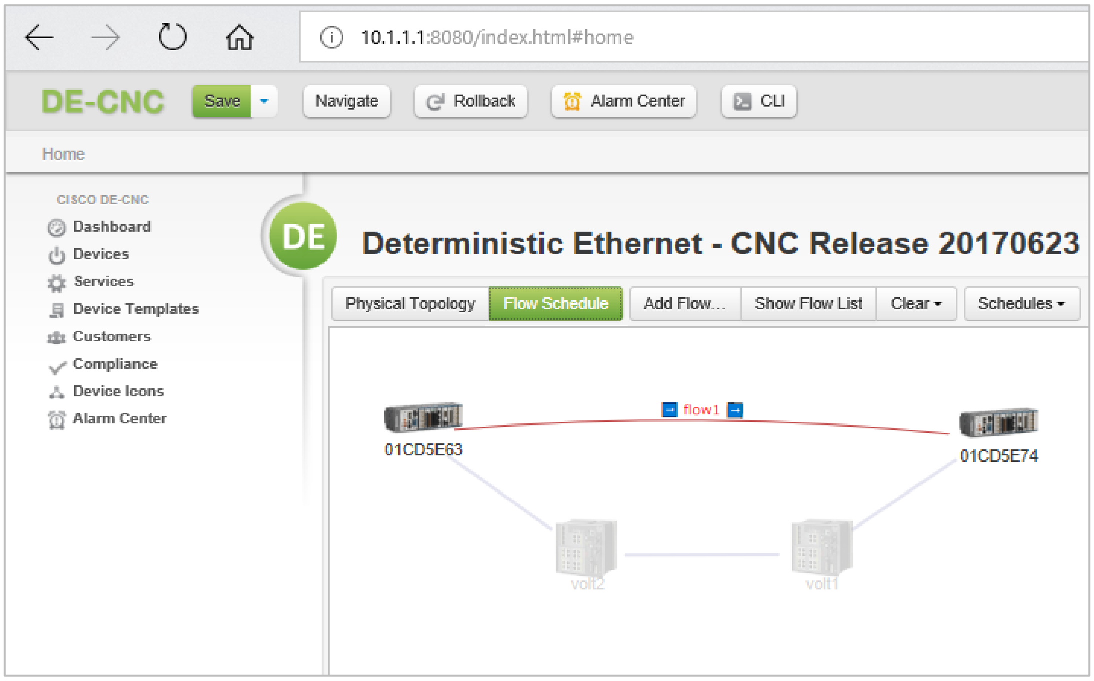
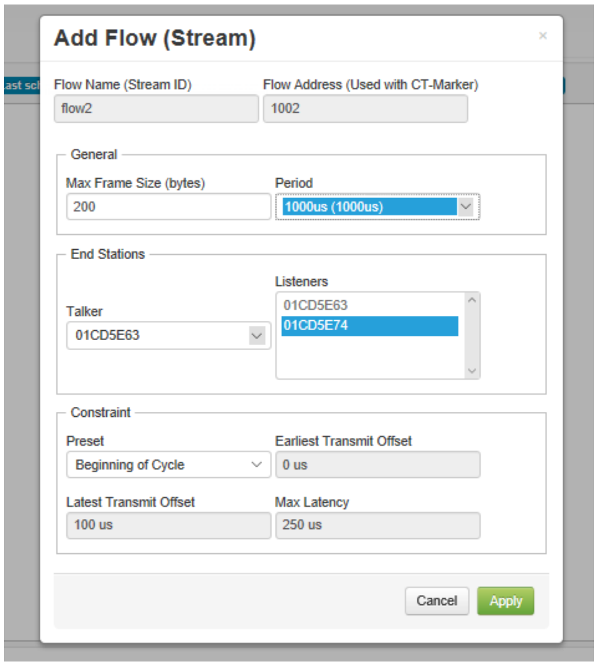


Figure 3.16: A scheduled 802.1Qbv TSN flow from talker to listener.

period of the flow is set (in the case of this test, the period was set to 1ms). Note that there appears to be a bug in the code that the actual period observed is double the period input, as a 1 ms period appeared to conform to a steady 2 ms of latency. In addition, a 500 µs period was observed as 1ms of latency.

After setting the period and the designated endpoints, the portion of a single period that the designated frames are sent is configured. This can be configured for the beginning, middle, or end of period. In addition, this can be configured manually. For example, given a 1000 µs period, the talker will send its data between 100 µs to 250 µs from the start of a period, with a given max latency and with respect to the start of the period.

After these settings have been configured and saved, the CNC must calculate the network wide flow schedule given the specified flows input. This may take some time given the computer hardware, the size of the TSN network, and the number of flows to be scheduled. After the schedule has been calculated, it must be distributed to all switches on the TSN network. Unlike AVB, TSN network determinism is administered via the bridges/switches, and not by the end points. After the switches have been updated via the


Add Flow (Stream)

Flow Name (Stream ID): flow2

Flow Address (Used with CT-Marker): 1002

General

Max Frame Size (bytes): 200

Period: 1000us (1000us)

End Stations

Talker: 01CD5E63

Listeners: 01CD5E63, 01CD5E74

Constraint

Preset: Beginning of Cycle

Earliest Transmit Offset: 0 us

Latest Transmit Offset: 100 us

Max Latency: 250 us

Figure 3.17: 802.1Qbv stream configuration.

CNC to the latest version of the TSN network schedule, the job of the CNC is finished. The switches all have the same schedule of when to let specific traffic through the network, based on the common perception of time via 802.1AS. The CNC is no longer needed unless changes to the network are required that necessitate an update to the switches schedules (such as new flows, new devices, or changed parameters of existing flows).

Note that the CNC software as utilized in the testing performed as part of this thesis is in beta release, and more features or configuration options may eventually be included as part of a final product.

3.4 Testing and Measurement Setup

This section will outline the steps taken in the testing of the TSN capabilities. The results of these tests will be presented and discussed in Chapter 4.

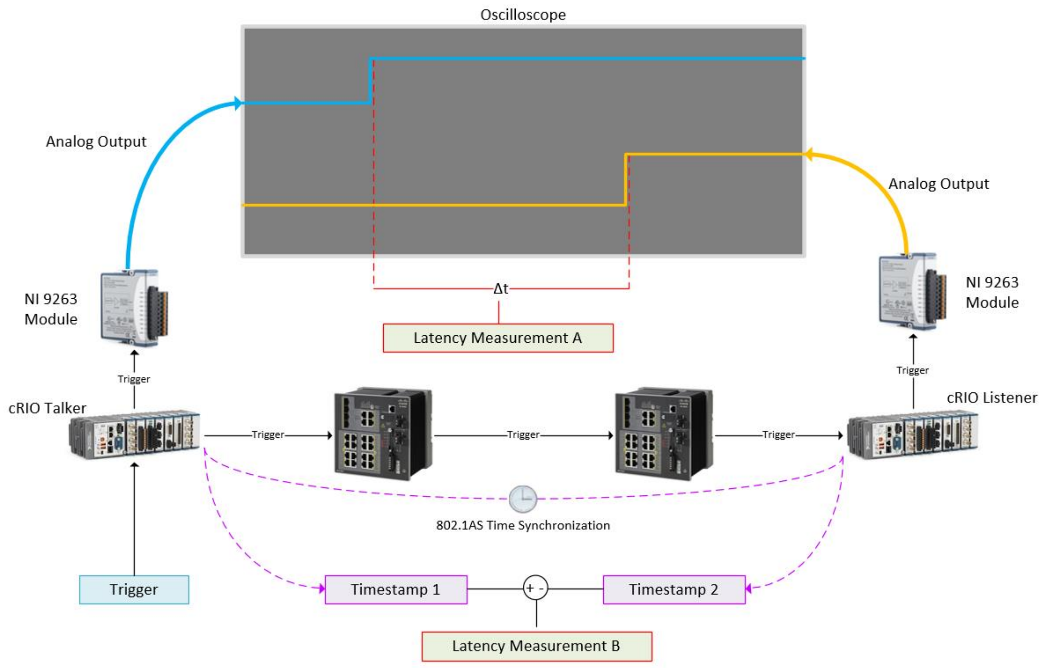


Figure 3.18: Latency measurements based on timestamp and oscilloscope data.

3.4.1 802.1Qbv (Latency) Testing

The outline for setting up 802.1Qbv requires more preparation and configuration than 802.1AS on the Cisco-cRIO platforms being used. Creating and configuring an 802.1Qbv TSN flow requires setting up the flow in the Cisco CNC, as well as in the Compact RIOs themselves.

To facilitate the accurate testing of the latencies from one cRIO to another, it was desirable to utilize the synchronized time measurement abilities afforded by 802.1AS to speed up testing. To this end, the first tests conducted were to ensure that using the hardware timestamps in place of oscilloscope measurements was an accurate way to measure latencies. To do this, it was necessary to implement a method of latency measurement that was independent of the communication protocol that was being tested. Code was written that (on the first cRIO) fed a trigger variable into the communication chain at the same moment that it was fed into the cRIO FPGA. FPGA code was written that would save the current time whenever this variable transitioned from 0 to 1. This same code was present on the second compact RIO. As the communicated data exited the

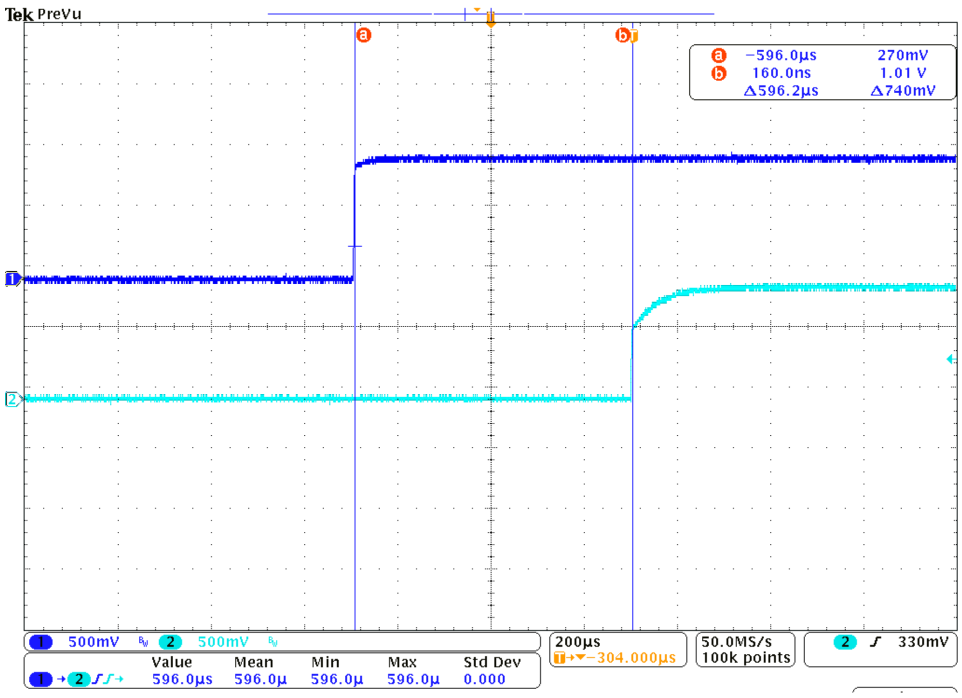


Figure 3.19: Latency measurement using cRIO FPGA analog output waveforms.

communication channel on the second cRIO, this variable was again fed into the FGPA module, and monitored for transitions. As noted previously, due to the loop delay in passing the trigger from the communication VI to the FPGA VI, all latency measurements (via timestamp or oscilloscope) suffer an additional 100 µs delay relative to when they truly arrived. Within the FPGA module of both devices, this variable was also output to the analog output module, for oscilloscope measurement. The first goal was to ascertain if the latency measured from the oscilloscope (from the moment cRIO1's analog output when high to the moment cRIO2's output went high) was in accordance with the same measurement based on 802.1AS synchronized timestamps.

The results of this test (shown in table Table 3.1) demonstrate that the measurement accuracy is accurate to within single digit microseconds. It should be noted that the timestamps are in the Unix time format, but due to an excel limitation allowing only 12 significant digits per cell, the first 4 digits of the timestamp are truncated to allow the significant portions of the timestamp to be entered. As the different between the two timestamps is immediately taken, this does not affect results.

Table 3.1: Oscilloscope and timestamp latency measurement verification data.

Trial	Talker Timestamp	Listener Timestamp	Timestamp Delay (ms)	O-scope Delay (ms)	Diff (ms)	Abs(Diff) (ms)	Diff (μs)
1	469944528877605	469944530398680	1.5211	1.5230	-0.002	0.002	2
2	469976143379955	469976144280905	0.9010	0.9012	0.000	0.000	0
3	470023959579430	470023960386230	0.8068	0.8078	-0.001	0.001	1
4	470049360089780	470049361178755	1.0890	1.0880	0.001	0.001	1
5	470056936189355	470056938089705	1.9004	1.9020	-0.002	0.002	2
6	470063867980305	470063868482680	0.5024	0.5053	-0.003	0.003	3
7	470071292182380	470071292684155	0.5018	0.5024	-0.001	0.001	1
8	470077739694630	470077740883830	1.1892	1.1900	-0.001	0.001	1
9	470095363578855	470095364881130	1.3023	1.3030	-0.001	0.001	1
10	470099978177505	470099978793105	0.6156	0.6180	-0.002	0.002	2
11	470106540778155	470106541488180	0.7100	0.7114	-0.001	0.001	1
12	470112322979280	470112324019855	1.0406	1.0420	-0.001	0.001	1
13	470116917577780	470116918478030	0.9003	0.9012	-0.001	0.001	1
14	470122845287180	470122845874005	0.5868	0.5879	-0.001	0.001	1
15	470127911896755	470127912681430	0.7847	0.7857	-0.001	0.001	1
16	470133733699555	470133734780430	1.0809	1.0830	-0.002	0.002	2
17	470139629281305	470139630241905	0.9606	0.9618	-0.001	0.001	1
18	470144850585030	470144851180180	0.5952	0.5960	-0.001	0.001	1
19	470150645083605	470150646389230	1.3056	1.3060	0.000	0.000	0
20	470155676185880	470155676723280	0.5374	0.5383	-0.001	0.001	1
21	470160275992955	470160276778805	0.7859	0.7857	0.000	0.000	0
22	470164880582330	470164881709105	1.1268	1.1270	0.000	0.000	0
23	470172901612130	470172902493630	0.8815	0.8820	-0.001	0.001	1
24	470180330381155	470180330975830	0.5947	0.5960	-0.001	0.001	1
25	470185099180930	470185099786555	0.6056	0.6070	-0.001	0.001	1
26	470189712983680	470189713880230	0.8966	0.8957	0.001	0.001	1
27	470194471383555	470194472091980	0.7084	0.7088	0.000	0.000	0
28	470201163288955	470201163784505	0.4956	0.4971	-0.002	0.002	2
29	470206824084680	470206825582280	1.4976	1.4980	0.000	0.000	0
30	470211884481580	470211885078530	0.5970	0.5960	0.001	0.001	1

As can be seen in Table 3.1, the 802.1AS synchronized timestamps are in agreement with the measured oscilloscope data down to below 5 µs. As there is still a certain level of uncertainty at 1 µs precision, and the latencies being tested are on the order of milliseconds and greater, all measurements in Chapter 4 are truncated to the 10 µs place, as this is the smallest level of precision with 100% agreement between timestamp and oscilloscope measurement. In the context of power systems and microgrid applications (operating at 60 Hz), 10µs is more than adequate for the level of precision needed for the tests to be performed.

Verifying the timestamp precision with the oscilloscope allowed the ability for larger data sets to be taken for comparison than manually triggering multiple oscilloscope tests. Rather than triggering the oscilloscope one measurement at a time, the testing has been automated to periodically trigger measurements with no user input after initiation. These measurements are saved to a text file on the Compact RIO.

With this new information, code was added to the FPGA file that allowed for up to 50 timestamps to be saved, representing 50 latency tests. As mentioned before, the FPGA code used to measure the latency is independent of the communication protocol being tested. In the tests, TSN was compared with directly sending TCP and UDP packets with the raw data values.

Artificial network congestion was injected into the network in an attempt to induce higher latencies at higher levels of network usage. The command line tool iPerf was installed on two virtual machines, one of which was connected to each of the two IE4000 switches. One iPerf instance acted as a server, receiving data from the other which acted as a client. The client was configured to send a UDP stream of a certain bandwidth the server. This stream traveled along the trunk connection between the two switches, which is limited to 1000 mbps line speed. The bandwidth of the stream was varied from 0 mbps (no additional congestion) to 950 mbps (full congestion). It should be noted that while 1000 mbps is the theoretical maximum bandwidth of the trunk connection, the max achievable bandwidth observed in testing was 958 mbps. Thus, 950mbps, while lower than the theoretical line speed, represents “maximum” line congestion. Higher levels of additional congestion were intended to show that higher

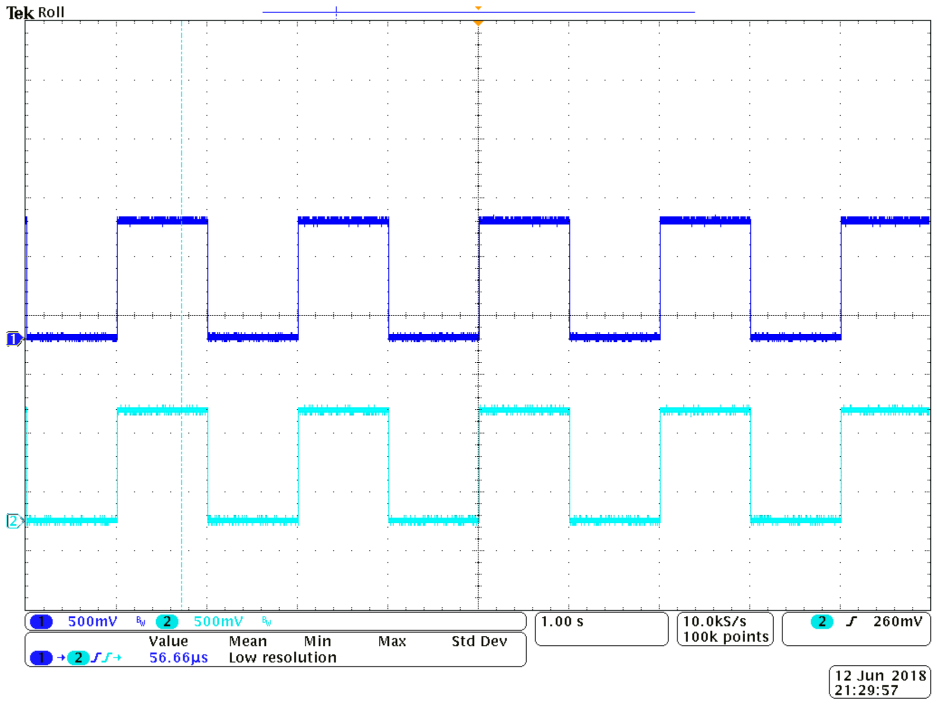


Figure 3.20: One pulse per second (1PPS) test to verify synchronization.

average latencies would be observed in the TCP/UDP cRIO communication protocols, while no inhibition would be observed in the TSN communication stream.

In addition to varying the protocol and line congestion, the size of the data packets sent was also varied. This was in an attempt to see if larger packet sizes would lead to higher overall latencies. To facilitate this, additional null data was added to the data cluster that was then sent to the communication protocol code for transmission. This data was not used for additional functionality, and was simply used to pad the data packets. The sizes tested were 20, 50, 100, and 200 bytes.

3.4.2 802.1AS (Time Synchronization) Testing

The timing synchronization features of IEEE 802.1AS present on the National Instruments Compact RIOs, unlike the other features such as 802.1Qbv, are for the most part automatic. The TSN network automatically chooses one of the two cRIOs to act as the grandmaster clock, and the timing synchronization is handled from there. This

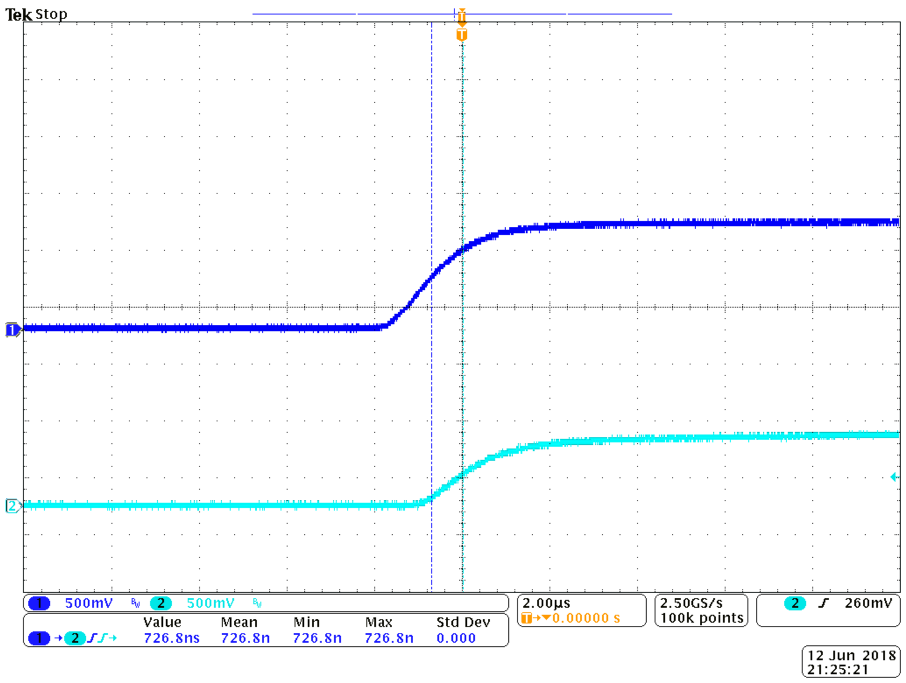


Figure 3.21: 1PPS test waveform at 2µs/div. Note Δt between rise times of 700 ns.

common perception of time can then be utilized in LabVIEW code as the programmer sees fit.

Before attempting to test synchronization of sine waveforms, an initial test to measure and verify the baseline performance of the timing abilities of 802.1AS was conducted. In this test, the cRIOs are configured to toggle their output between 0 and 1 every 1 second interval (measured in nanoseconds). The output was measured on an oscilloscope to verify the synchronization performance within 1 µs or better (as described by the 802.1AS standard). As shown in Figure 3.20, the outputs of both cRIOs toggle at every time value evenly divisible by 1,000,000,000 nanoseconds. By comparing the difference in the rise times, the precision of the clock synchronization can be observed. In this case, clock synchronization error averaged approximately 700 ns, better than the 1µs specified by the standard, but not as ideal as expected. Figure 3.21 shows a zoomed-in view of the 1PPS test.

The timing synchronization tests performed in this implementation include the generation of a synchronized “sync” waveform from the output of the cRIO FPGA to the analog output module installed in each device. This sync waveform would be controlled and synchronized solely via the common perception of time that both cRIO systems have because of the IEEE 802.1AS time synchronization present on the devices. The output waveform is generated from the FPGA of the cRIO, via information sent internally from the front panel control layer. At the front panel layer, a frequency setpoint is specified. The LabVIEW code then converts this setpoint from a frequency to a period in nanoseconds. In addition, the current time is noted and summed with a second user-configurable variable called *Response Time Setpoint* to obtain a *Start Time* for the new frequency output. The *Response Time Setpoint* variable governs how much time is elapsed between the changing of the frequency setpoint, and when the actual new frequency output of the FPGA is to take effect. For example, if *Response Time Setpoint* is set to 10ms, and the output frequency setpoint is changed by the user, the cRIOs will begin outputting the new frequency setpoint in unison 10 ms after the user alters the setpoint.

These two variables, the period of the frequency setpoint and the new *Start Tim*e variable, are then passed to the FGPA on the local unit. The same variables are also sent over the network to the second cRIO, which passes the same information to its own FPGA. The FPGA continues outputting the previous frequency setpoint value until the new *Start Time* is reached. At this time, the FPGA begins outputting the new frequency setpoint. Due to the FPGAs common perception of time, this transition should happen at the same moment on both devices. In addition, the frequency update may be continuous and real-time, allowing both units to output synchronized arbitrary sine waveforms with varying frequencies.

The *Response Time Setpoint* mentioned above changes the time buffer for the new update time and frequency setpoint to make it from the first compact RIO to the second. For a TSN based communication system with minimal latency (1-2 ms), the frequency update can arrive before the end of a single cycle, allowing near real-time levels of frequency control. UDP potentially has the ability to come close in terms of speed, and will be tested as well to see if synchronization can be maintained with high congestion. Other protocols like TCP cannot guarantee that the update will reach the second cRIO within a certain period of time, and thus to avoid the new Start Time update occurring

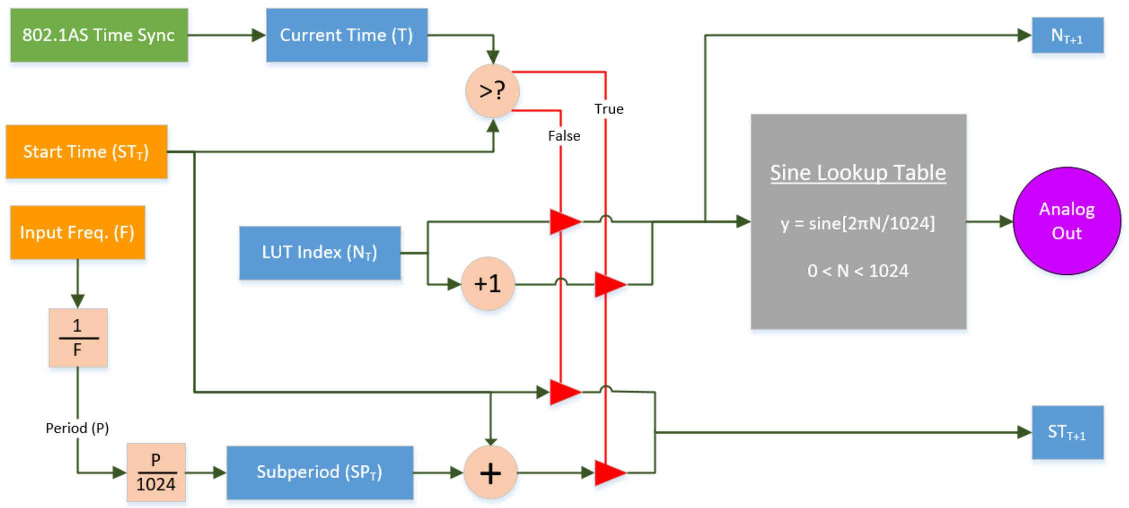


Figure 3.22: Simplified block diagram of synchronized waveform generation using 802.1AS. Orange boxes are data variables passed from the main VI into the FPGA VI.

before the command has arrived at the second compact RIO, large values for *Response Time Setpoint* must be used. If the update arrives after *Start Time*, the synchronization of the output waveforms would be thrown off. This also means that any desired change must be given 2-5+ seconds of buffer to take effect, eliminating any real-time control potential of the protocol.

In the tests conducted in Chapter 4, the synchronization precision over time is measured for both static (60Hz) and time-varying frequency setpoints (shown in Figure 3.23). With a *Response Time Setpoint* below 16 ms, the goal is to demonstrate a signal synchronization (within 1 degree of phase shift) while updating frequency setpoint within a single cycle. This is demonstrated in both a low congestion network, and a maximally congested network for the UDP and TSN protocols.

3.5 Chapter Summary

In Chapter 3, the equipment used for testing and characterizing TSN features in a network were introduced. Two NI Compact RIOs serve as the TSN endpoints, with two Cisco IE4000 switches acting as TSN bridges.

The testing plan for generating network congestion to better emulate a large high traffic network system was also detailed. Utilizing the iPerf network utility, real network

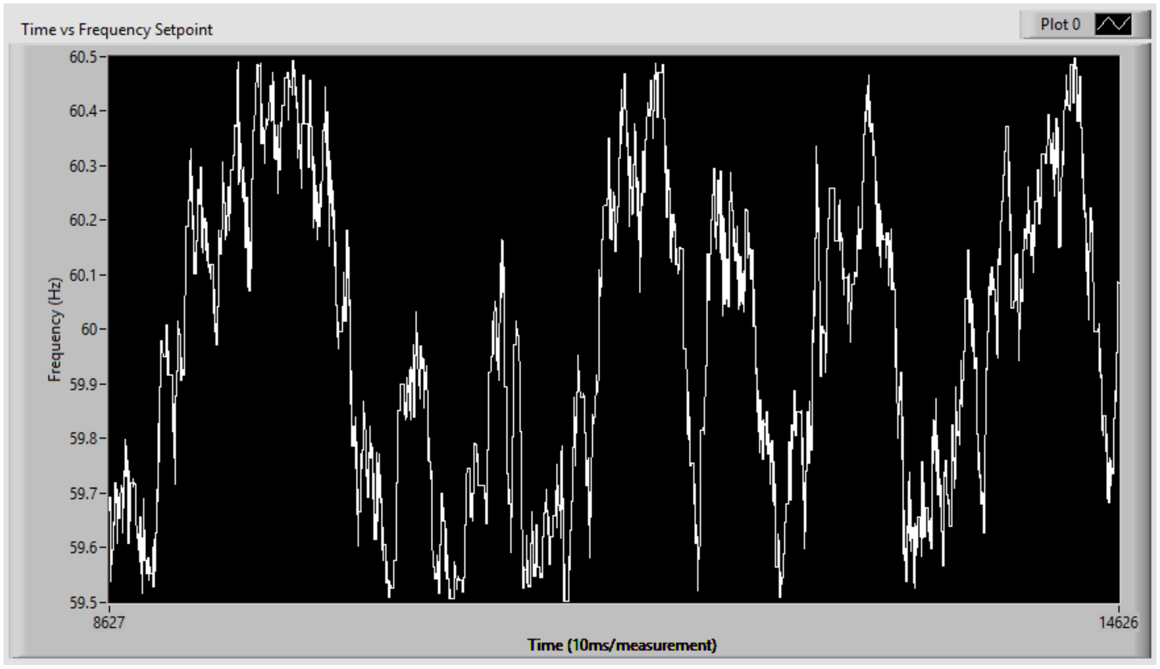


Figure 3.23: A time-varying frequency setpoint used to determine synchronization performance. The width of this graph represents 60 seconds.

congestion can be generated to provide consistent network inhibitions for comparison. This congestion can be tuned to inhibit the network to varying degrees, to emulate different levels of network impairment.

The first test to be performed in Chapter 4 is to characterize the latency of UDP, TCP, and TSN through networks of different congestion levels. Different sizes of data frames will also be examined, to determine if packet size greatly affects latency results. It is expected that while UDP and TCP will experience (on average) larger latencies at higher congestions and larger data frames, TSN will maintain a consistent latency regardless.

The second test to be performed is experimentation with generating of synchronized sine waves from each cRIO. Using the latency features of TSN, the potential for real-time frequency tracking with sub-cycle update times will be explored. The TSN case will be compared with the UDP case, as UDP (which is optimized for speed) is the natural competitor to TSN for this type of application. By alternating the frequency setpoint randomly within a specified range, and observing the phase shift between the two output waveforms, the feasibility of this method will be examined.

4 RESULTS AND DISCUSSION

4.1 NETWORK LATENCY PERFORMANCE – UDP

The results of the latency testing of the UDP protocol over a highly congested trunk link are presented in Figure 4.1 through Figure 4.4. Each graph represents a different data frame size. Each column of data represents latency measurements from 250 trials. The columns represent 0%, 50%, 75%, 85%, 90%, and 95% capacity (or congestion) of the trunk link connecting the two IE4000 switches used for testing. The data is presented in a box and whisker plot. The same graph format is presented for both the TCP and TSN test cases.

It should be reiterated that UDP is a protocol that prioritizes speed over reliability. Thus, the expectation was that while higher levels of network congestion and frame size would lead to higher overall latency, the rise would be relatively small. The mean/median latency values rise slowly as congestion increased in the trunk line, though high levels of congestion clearly show greater numbers of high latency packets overall. Though the highest latency packets observed are statistical outliers from the main bulk of the measurements, their presence is still significant. Though they are few, a non-zero number of high latency packets could potentially cause decreased performance in a critical application that requires tight latency for the delivery of data.

In addition, it can be observed in Figure 4.1 through Figure 4.4 that for a given congestion level, as the size of the UDP frame increases, the large block (second and thrd quartile) where most of the latencies measurements fall rises. However, it should be noted that at 90% and 95% for a 200 byte frame size, the average latency dropped slightly. No root cause was determined, but the tests were still considered valid, as the overall trend of higher congestion causing higher latencies for a given frame size was still observed. In addition, the average of the highest 10% of latency measurements at this condition still increased, as shown in Figure 4.6.

Figure 4.5 through Figure 4.7 compare the results for each of the different UDP frame sizes together in a single graph. From the graphs, it can be seen that the size of the data frame transmitted can affect the latency experienced by the packet. This is especially evident when observing only the top 10% of packets with the largest observed latencies. This equates to the largest 25 latency samples for each 250 sample test. However, even at the highest levels of network congestion, very few latency measurements exceeded the length of a single 60 Hz electrical cycle. However, considering that these measurements were taken over a single congested link between two switches, latency would likely grow with more hops.

UDP, due to its high speed but low reliability, is best used in applications with streams of data that don't need to be delivered in order or error free. For example, a Voice over IP (VoIP) phone call needs to be fast, but a few dropped packets (leading to the occasional garble) is acceptable. However, a power system or microgrid application using UDP might include streaming of measurement data from a distributed sensor or a setpoint that only needs to include the most up to date value. Since UDP packets are likely to include some losses and out of order deliveries, they are not best suited for applications that require high reliability for every data point.

4.2 NETWORK LATENCY PERFORMANCE – TCP

TCP, unlike UDP, is designed for higher levels of reliability. While UDP simply streams one-way data into the network, TCP requires that a response message confirming delivery of each packet. TCP uses this functionality to detect and resend lost packets after a period of time or if packets sent after the lost packet are confirmed delivered (with the *ACK* message). When lost packets are detected, the rate of packet transmission is slowed, to prevent sending more packets likely to be lost into the network until the congestion has cleared. In addition, even during normal operation, there is a limit to how many data packets TCP sends into the network before waiting to confirm more data has been received at the destination. Due to these factors, it was anticipated that TCP would demonstrate much greater latencies on a congested network than UDP.

Figure 4.8 through Figure 4.11 show the measured latencies for TCP packets of various sizes and at various network congestion levels. Particular attention should be taken of the vertical scale of these graphs when compared to the UDP cases. The highest

TCP latencies measured during testing reached multiple seconds, unlike the tens of milliseconds observed the UDP test case. Much like the UDP tests, the latency measured when using TCP as the communication medium grew with both link congestion and data payload size.

Figure 4.12, Figure 4.14, and Figure 4.16 show the mean, mean of the top 10% and median latency values for all payload sizes. Due to the large variance of the latency values measured using TCP, Figure 4.13, Figure 4.15, and Figure 4.17 show the same data as Figure 4.12, Figure 4.14, and Figure 4.16 (respectively), using a logarithmic vertical scale.

Compared with UDP, TCP shows a much wider degree of change at higher congestion levels than UDP. Low level congestion latencies are similar to UDP, falling into the single digit millisecond range. However, at around 75-85% network congestion (depending on frame size) the latency begins rising dramatically, with average latency reaching 50-200 ms. This rise continued until max congestion, where the average latency is on the order of 1 second or more. At full congestion, some latency measurements reached as high as 5 seconds or more, beyond the scale of the graphs as presented. It is clear that TCP's reliability does come at the cost of its speed on a heavily trafficked network. As with the UDP test case, observed latency would be projected to increase further if more hops were added to the system, or larger data payloads were transmitted.

4.3 Network Latency Performance – TSN

As was alluded to in the comparison figures for both UDP and TCP, TSN latency has superior performance at all levels of network congestion. This is shown in Figure 4.18 through Figure 4.21. As expected, latency is deterministic and consistent as set by the user in the Cisco CNC. This is shown as a very flat, tightly packed bundle of latency measurements, regardless of congestion level. In addition, at all frame sizes, TSN latency continued to measure reliably at 2.1ms. Figure 4.23 and Figure 4.22 shown the mean and median values for the observed measurements for various payload sizes and congestion levels. As expected, these curves form a flat line, showing no apparent variance at any level of congestion or payload size.

The tests performed for TSN portion of this project was conducted to achieve a 2 ms latency. This was chosen to give a good balance of low latency (compared to TCP and UDP), while leaving enough computation resources for the cRIOs additional functions. In addition, other informal tests (not shown) were able to reduce this latency to approximately 1 ms by altering the period setpoint in the Cisco CNC. While it was possible to maintain these latencies, the computational requirement on the Compact RIO was higher than desirable, and other cRIO functions seemed to lag. For this reason, the initial 2 ms latency setpoint was tested, as this gave a good balance between fast latency for testing, while also leaving enough cRIO resources to run the additional functions. Lower latencies were attempted (500 µs), however the latencies measured were not stable, and varied by much higher degrees than the 2 ms and 1 ms test cases. Based on a presentation by National Instruments on the TSN implementation on cRIO devices, the period setpoint has a theoretical range between 200 µs and 1 second [42]. This limit may require the use of National Instrument's more powerful controllers.

As was noted previously in Chapter 3, there appears to be an issue where a 1 ms CNC latency setpoint in the user interface translates to a 2 ms observed latency (500 µs frequency setpoint was also observed as 1 ms of observed latency). As was noted previously, an additional 100 µs of delay from the communication function's loop iteration before passing the trigger variable to the FPGA VI can be observed in these results. As 1 ms was set in the Cisco CNC, 2 ms was the expected measurement, with an additional 100 µs from the communication loop. This consistent measurement at 2.1 ms is what was observed.

One point of note is that TSN cannot achieve lower levels of latency than is set in the CNC. It can be seen in the previous graphs for TCP and UDP that at lower congestion levels, many measurements are lower than the TSN case. Since TSN achieves its low latency by sending high priority traffic at a set loop interval, TSN will never achieve lower latency than the set loop interval. However, unlike TCP and UDP, the latency observed using TSN was far more consistent at every payload size and congestion level.

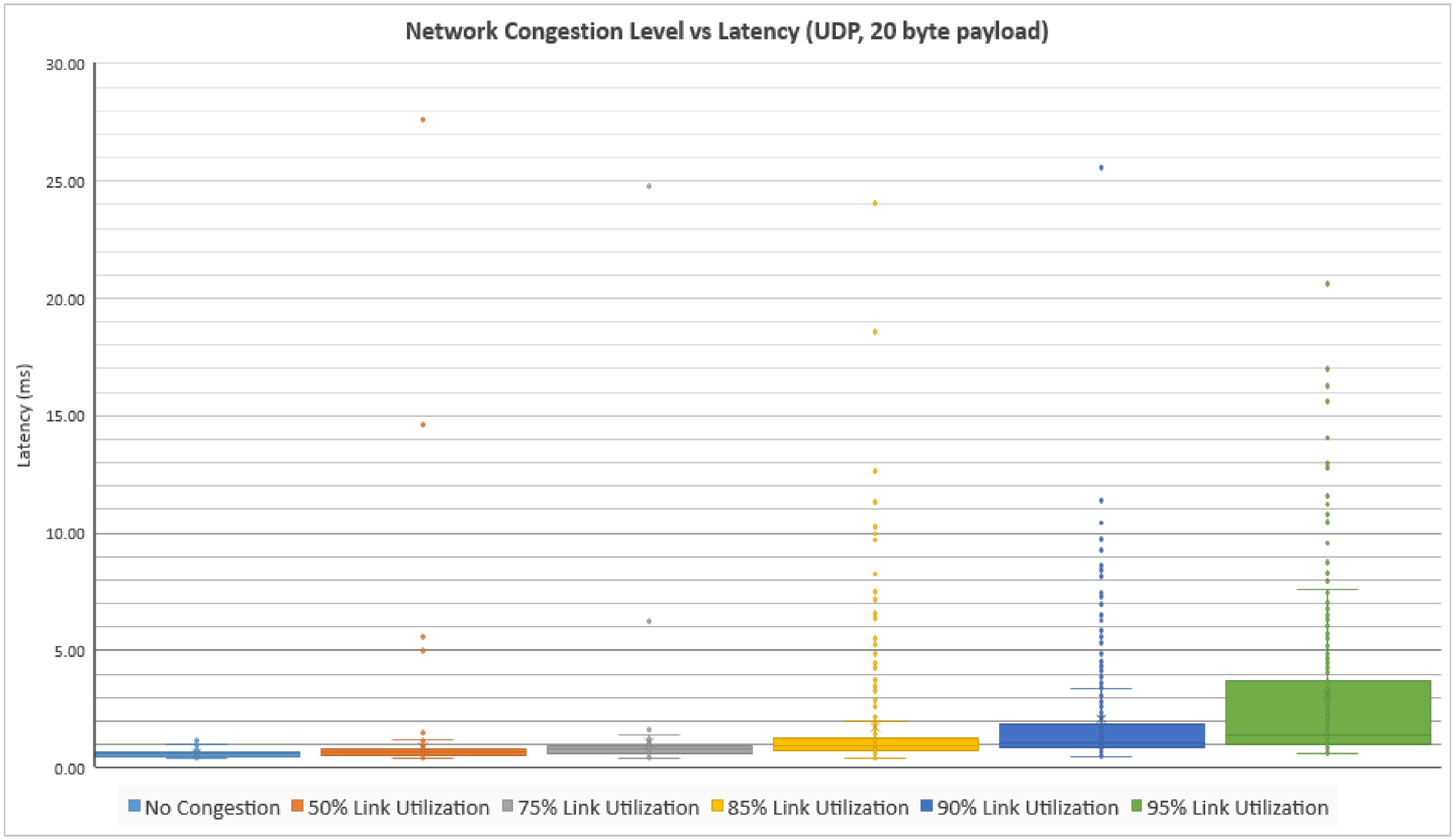


Figure 4.1: Distribution of network latency for a 20-byte UDP frame at different levels of network congestion.

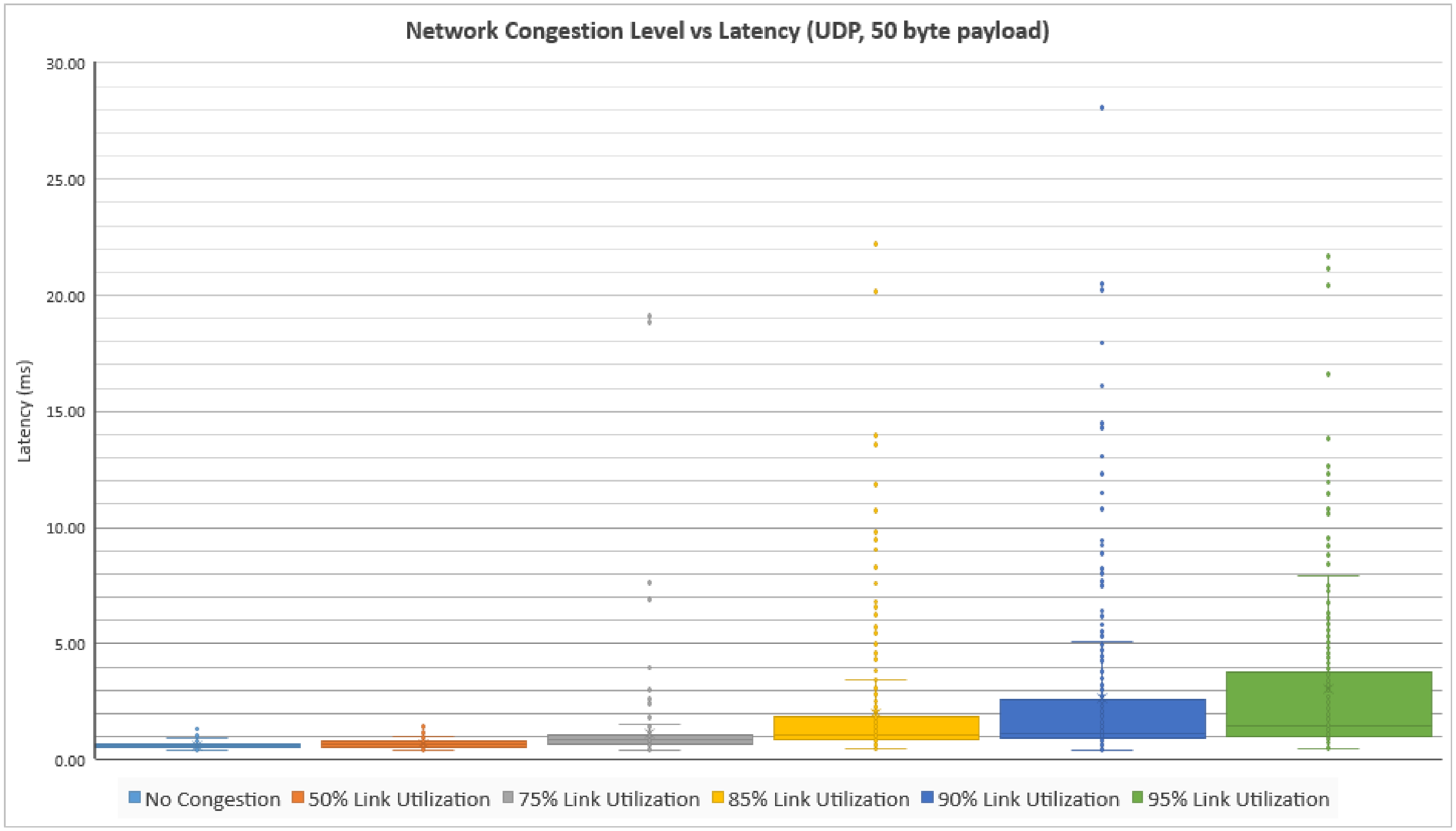


Figure 4.2: Distribution of network latency for a 50-byte UDP frame at different levels of network congestion.

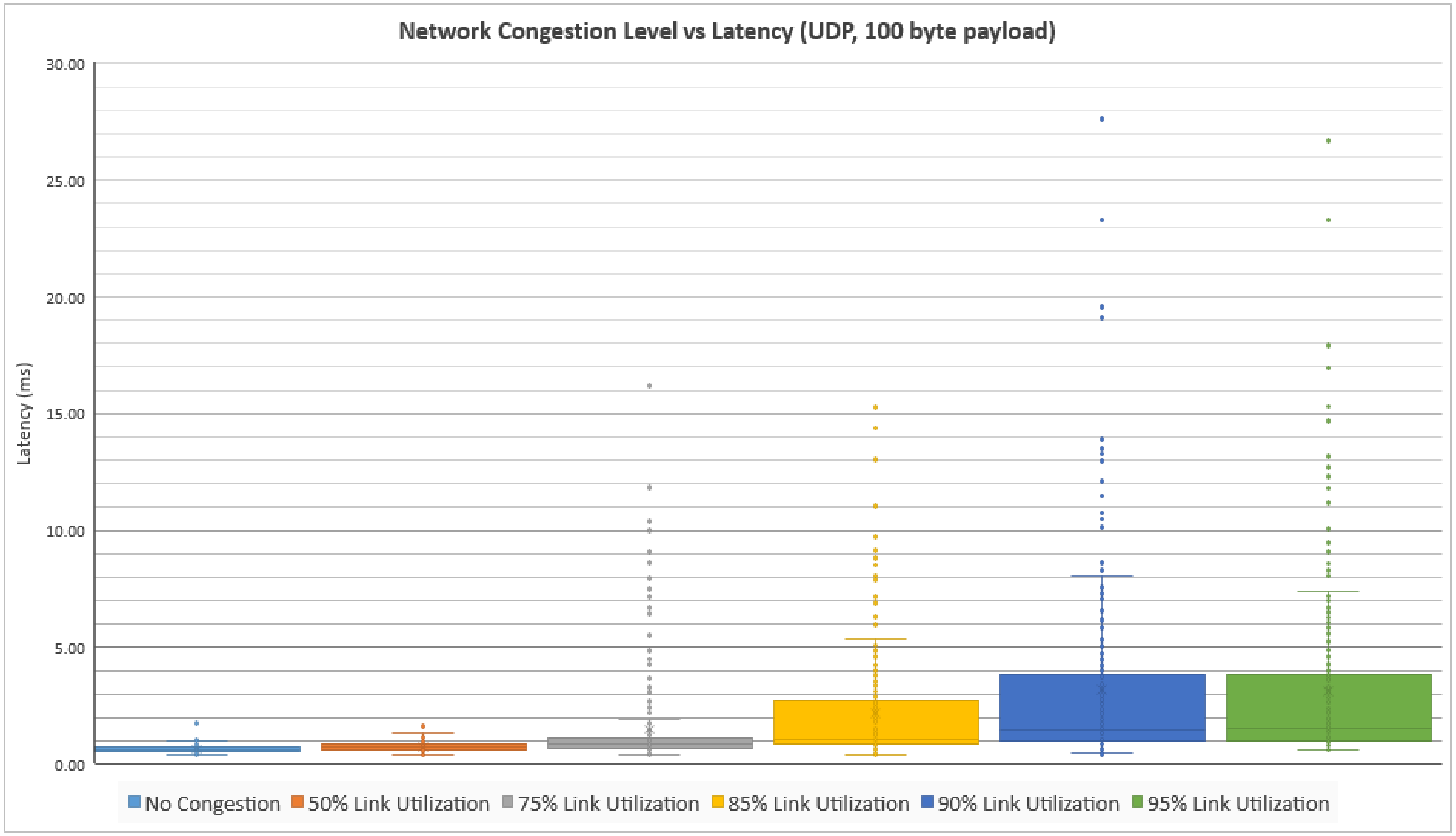


Figure 4.3: Distribution of network latency for a 100-byte UDP frame at different levels of network congestion.

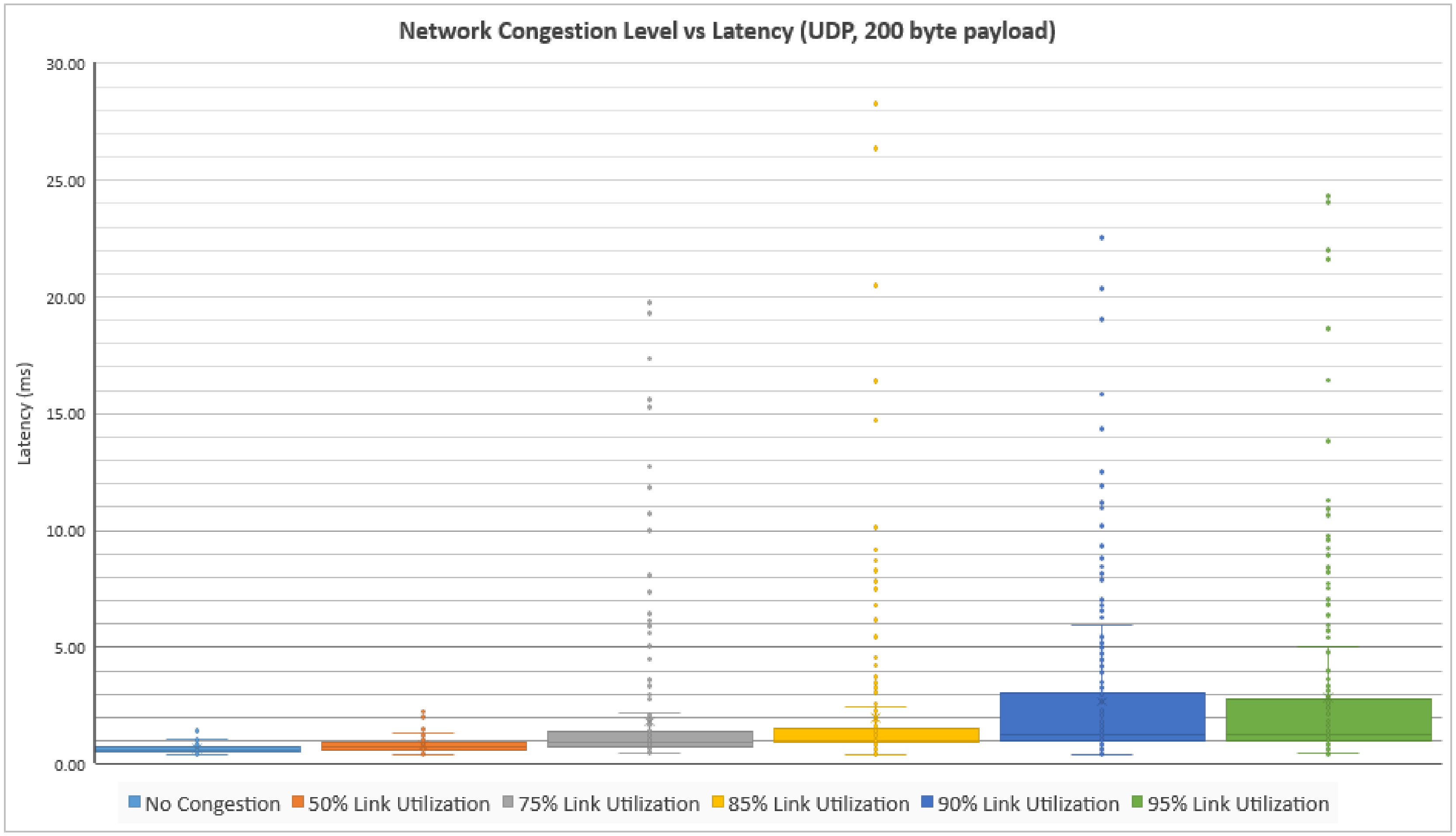


Figure 4.4: Distribution of network latency for a 200-byte UDP frame at different levels of network congestion.

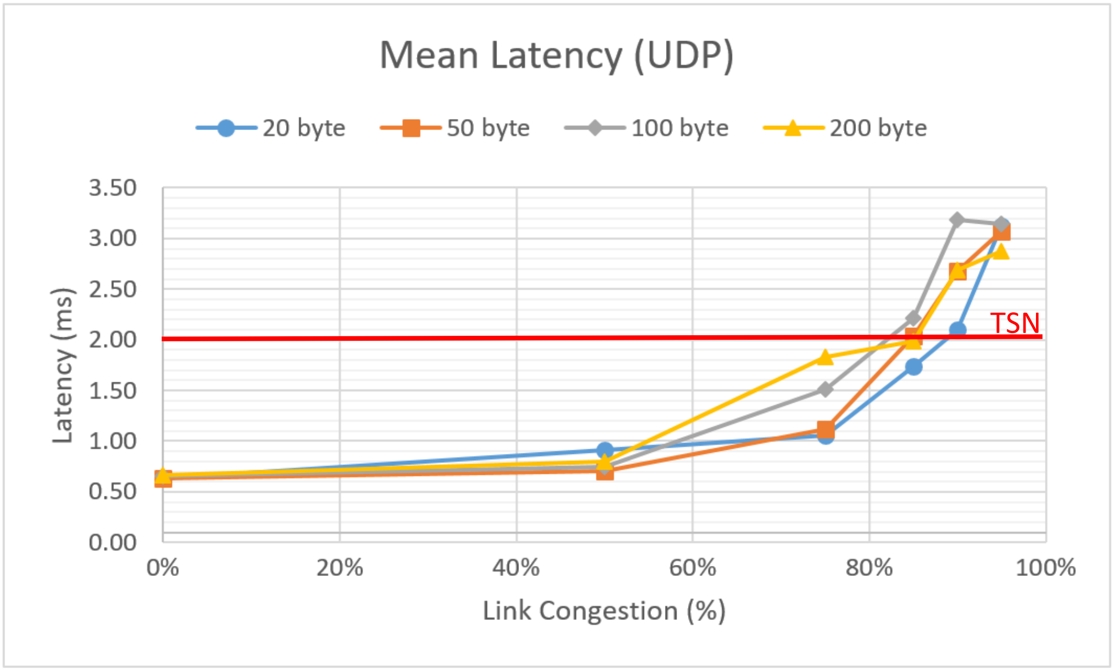


Figure 4.5: Average network latency vs congestion for different UDP frame sizes.

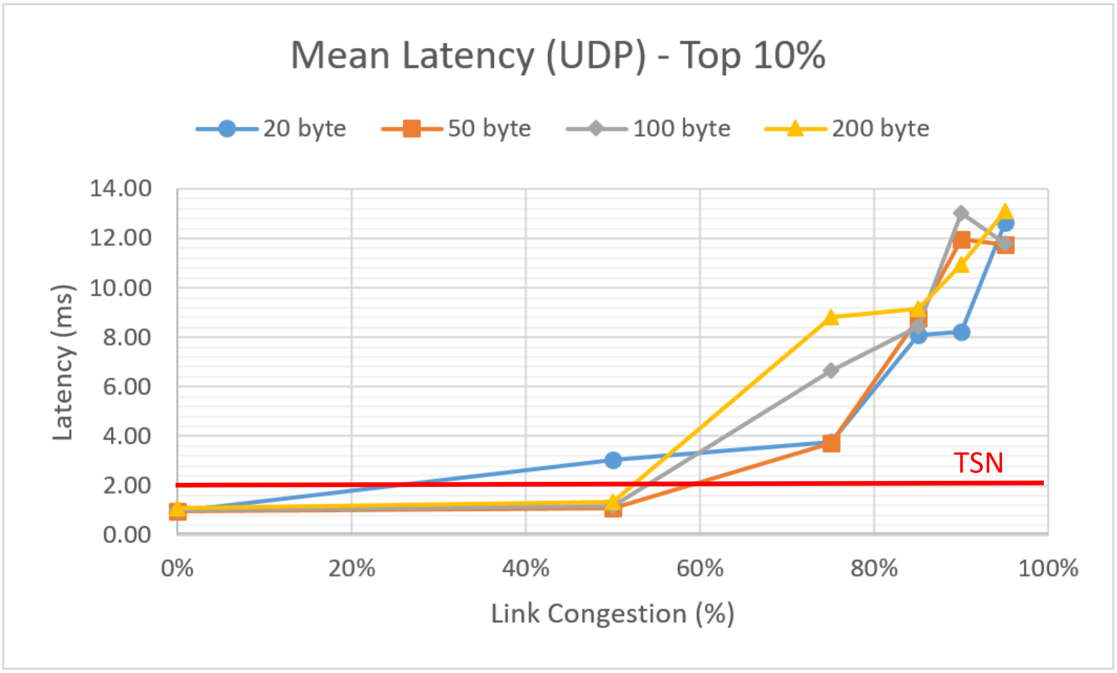


Figure 4.6: Average latency (top 10%) vs congestion for different UDP frame sizes.

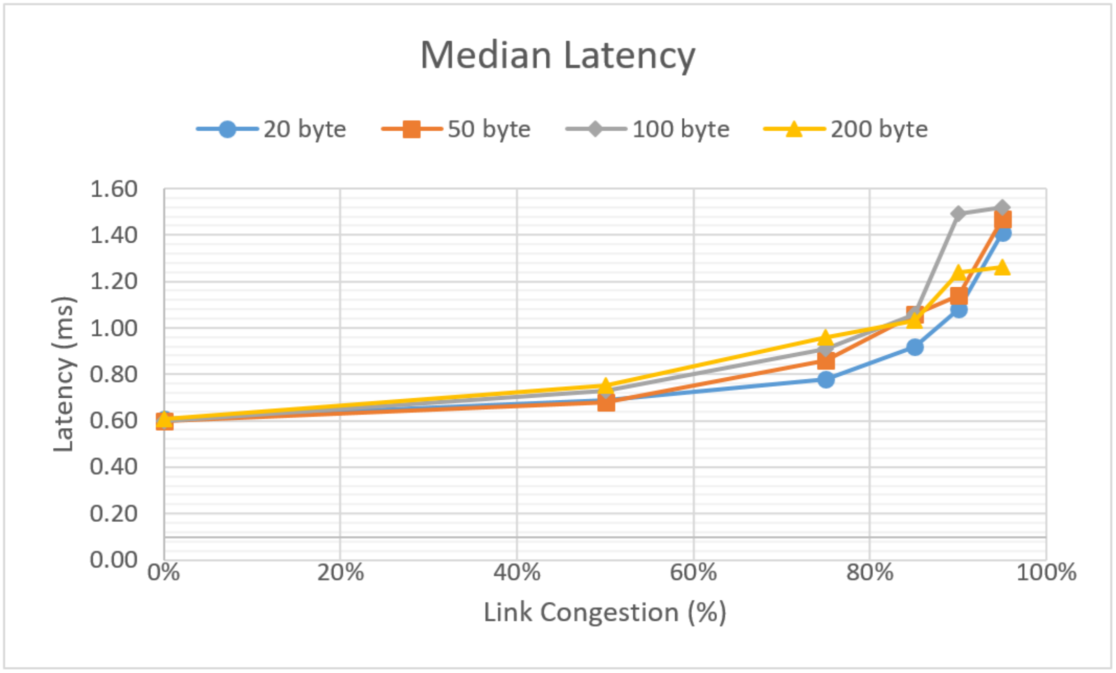


Figure 4.7: Median network latency vs congestion for different UDP frame sizes.

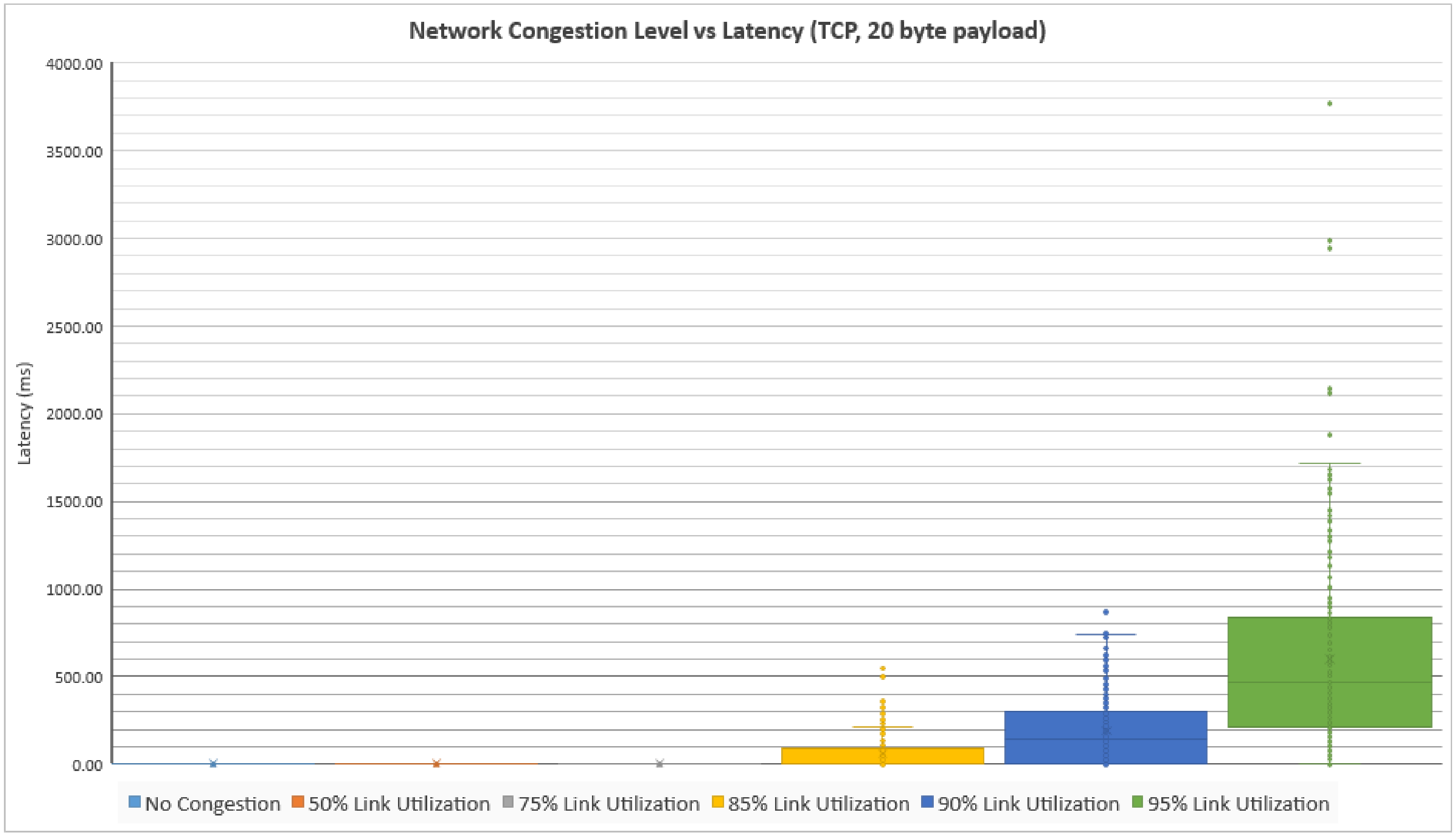


Figure 4.8: Distribution of network latency for a 20-byte TCP frame at different levels of network congestion.

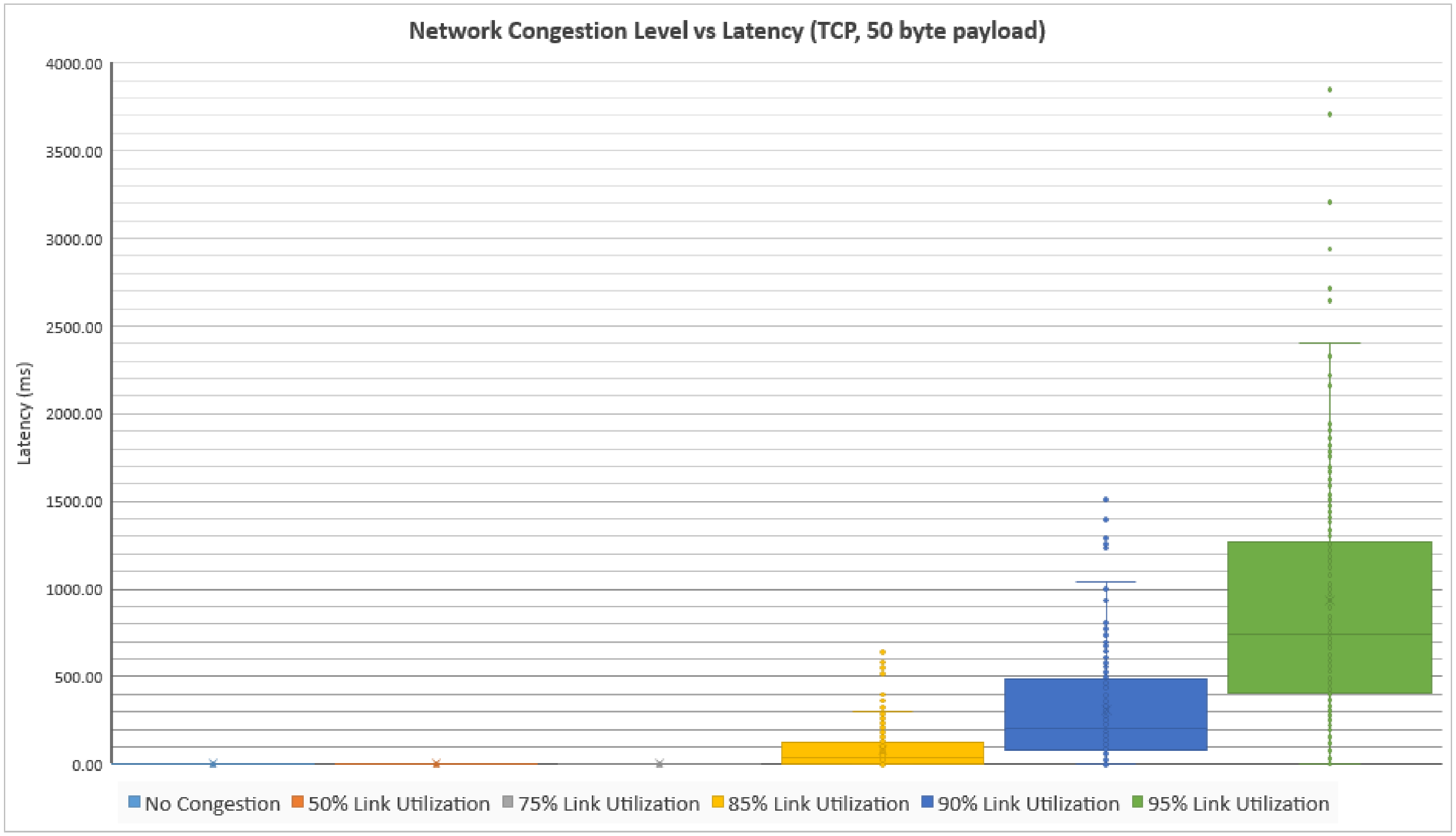


Figure 4.9: Distribution of network latency for a 50-byte TCP frame at different levels of network congestion.

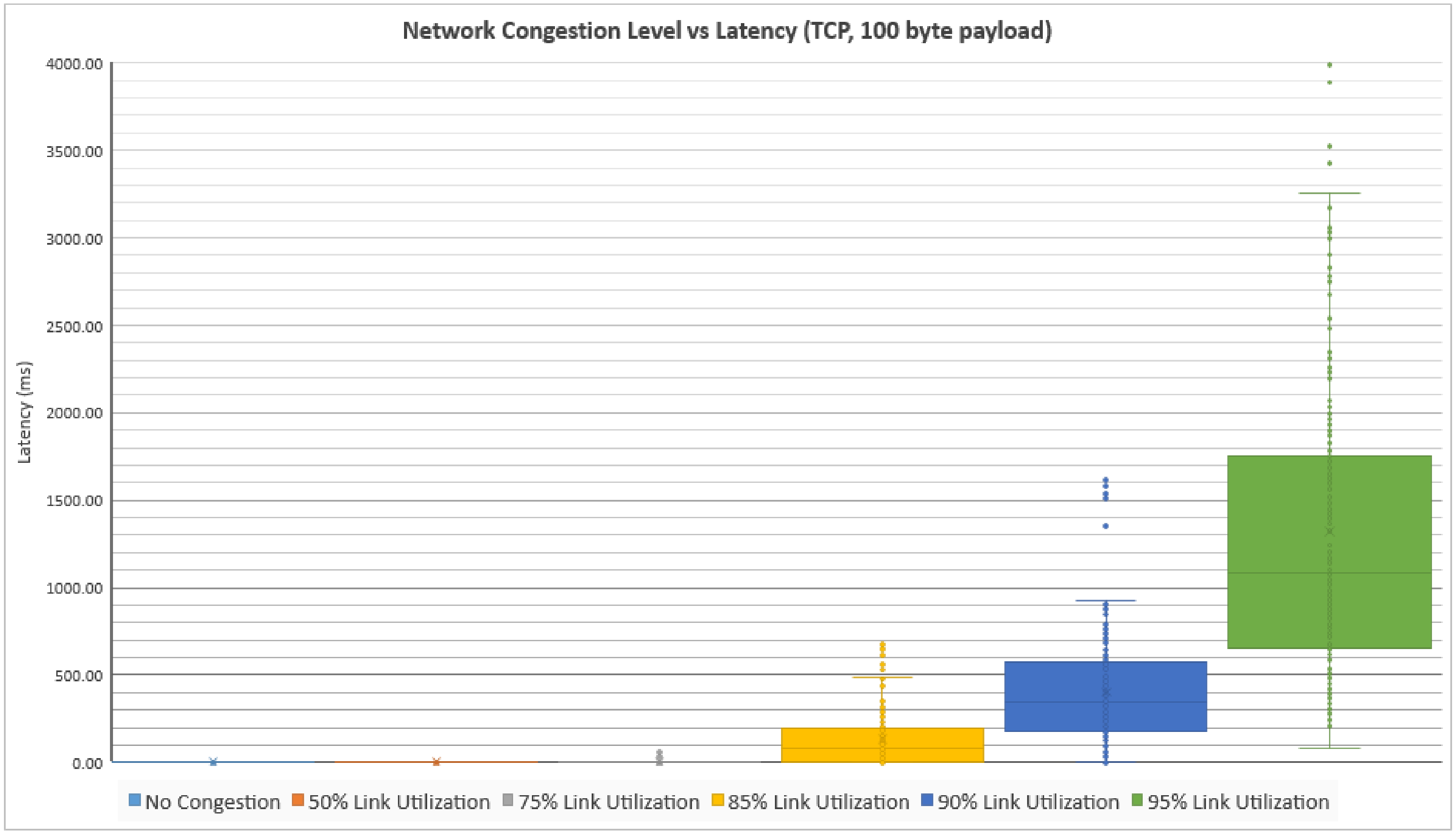


Figure 4.10: Distribution of network latency for a 100-byte TCP frame at different levels of network congestion.

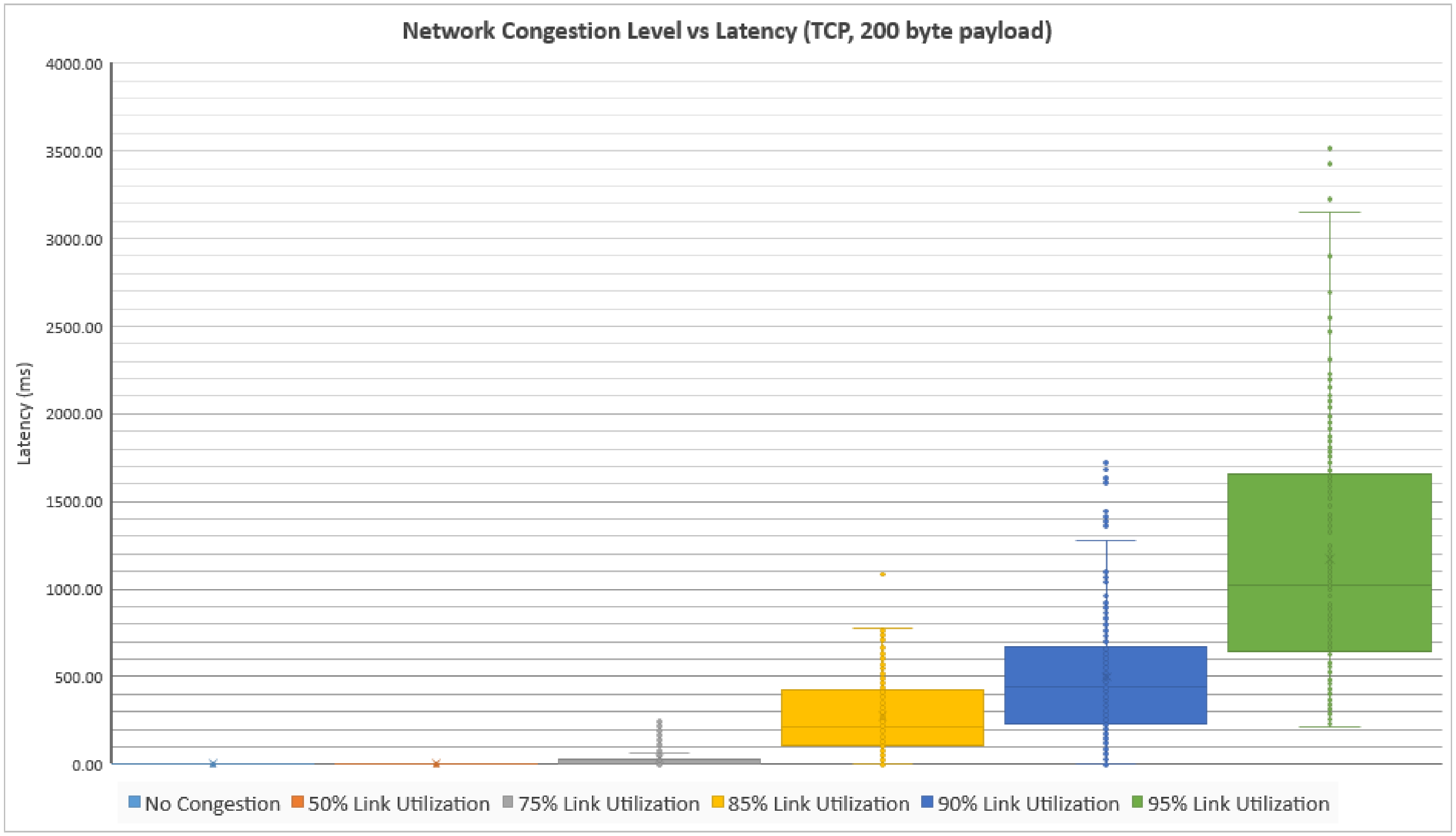


Figure 4.11: Distribution of network latency for a 200-byte TCP frame at different levels of network congestion.

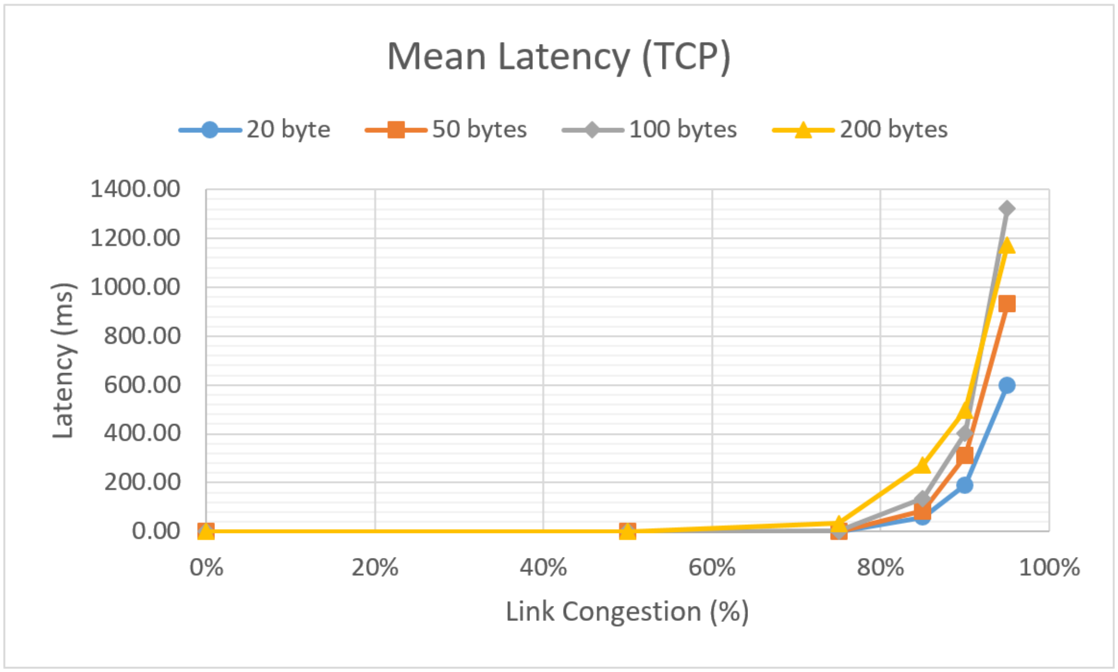


Figure 4.12: Average latency vs congestion for different TCP frame sizes.

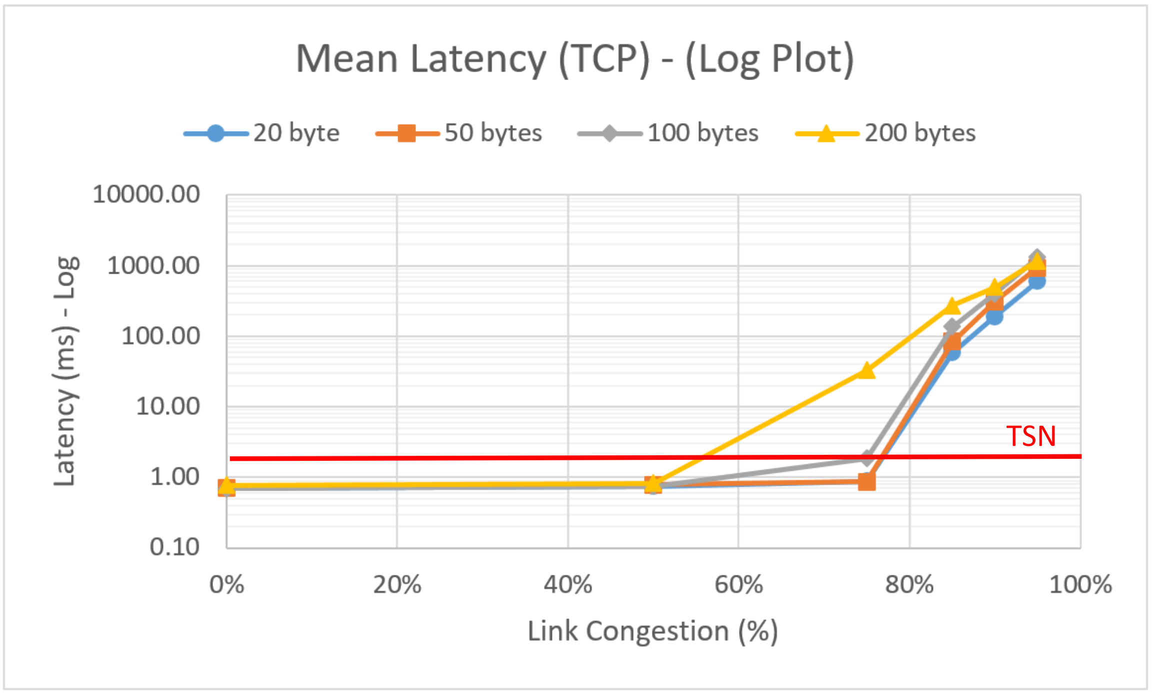


Figure 4.13: Average latency vs congestion for different TCP frame sizes – log plot.

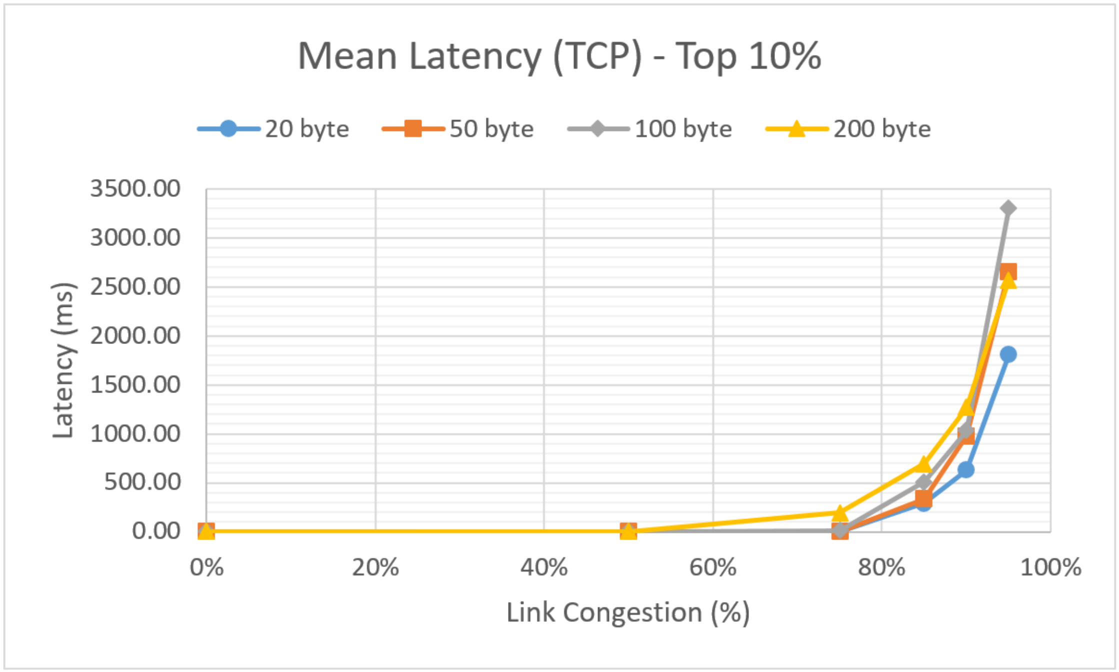


Figure 4.14: Average latency (top 10%) vs congestion for different TCP frame sizes.

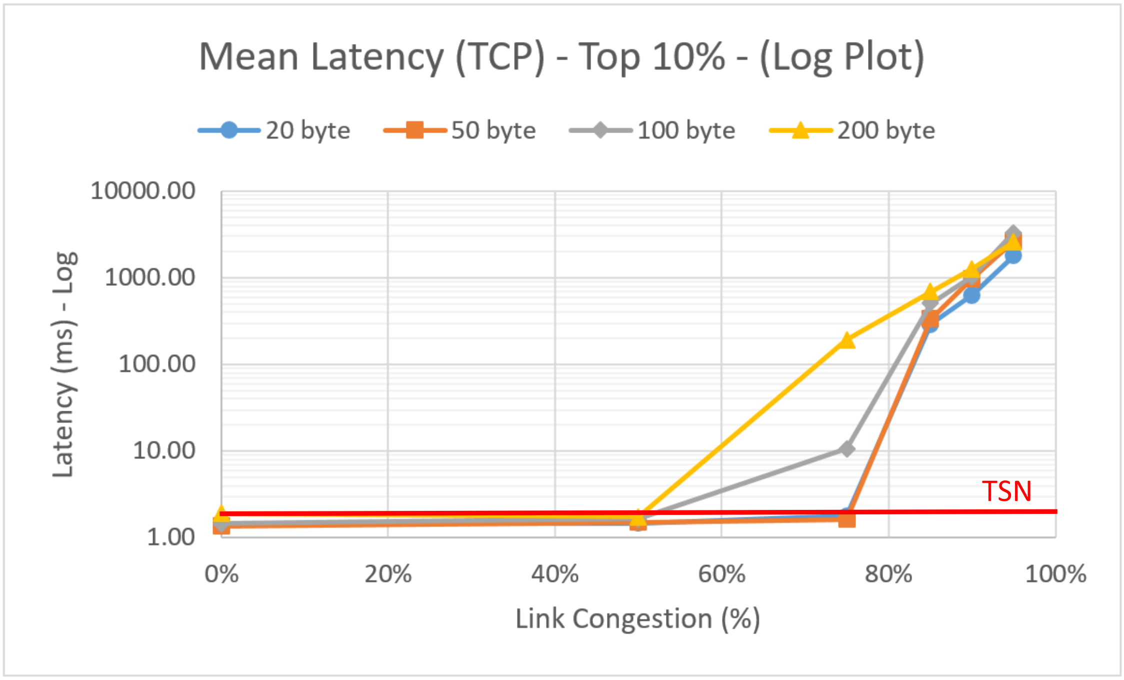


Figure 4.15: Average latency (top 10%) vs congestion for different TCP frame sizes – log plot.

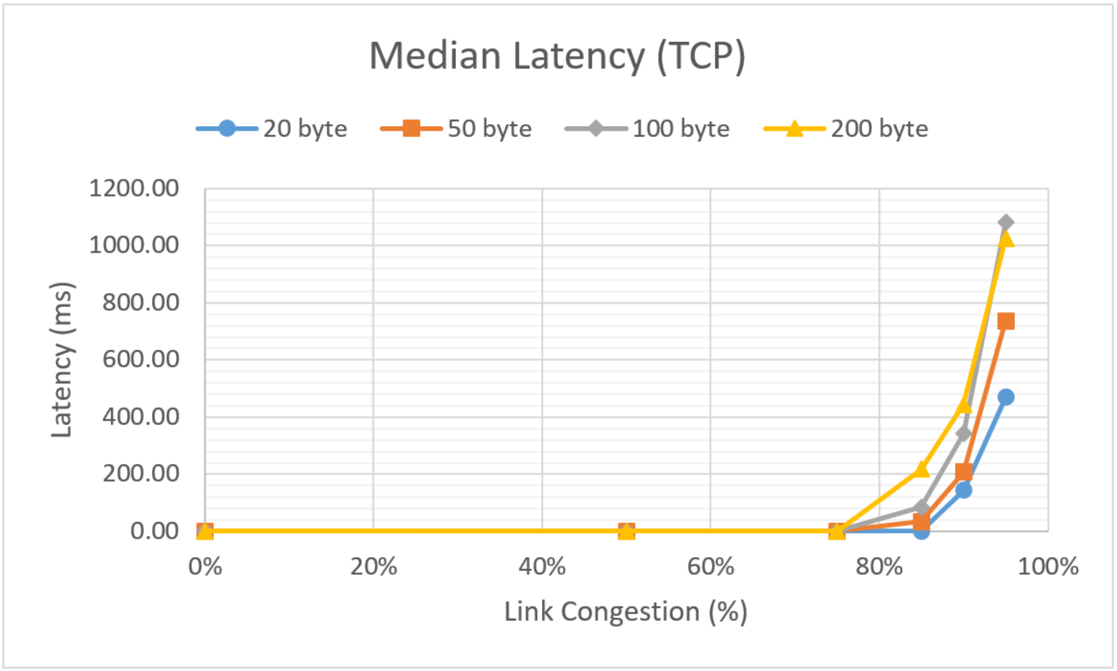


Figure 4.16: Median latency vs congestion for different TCP frame sizes.

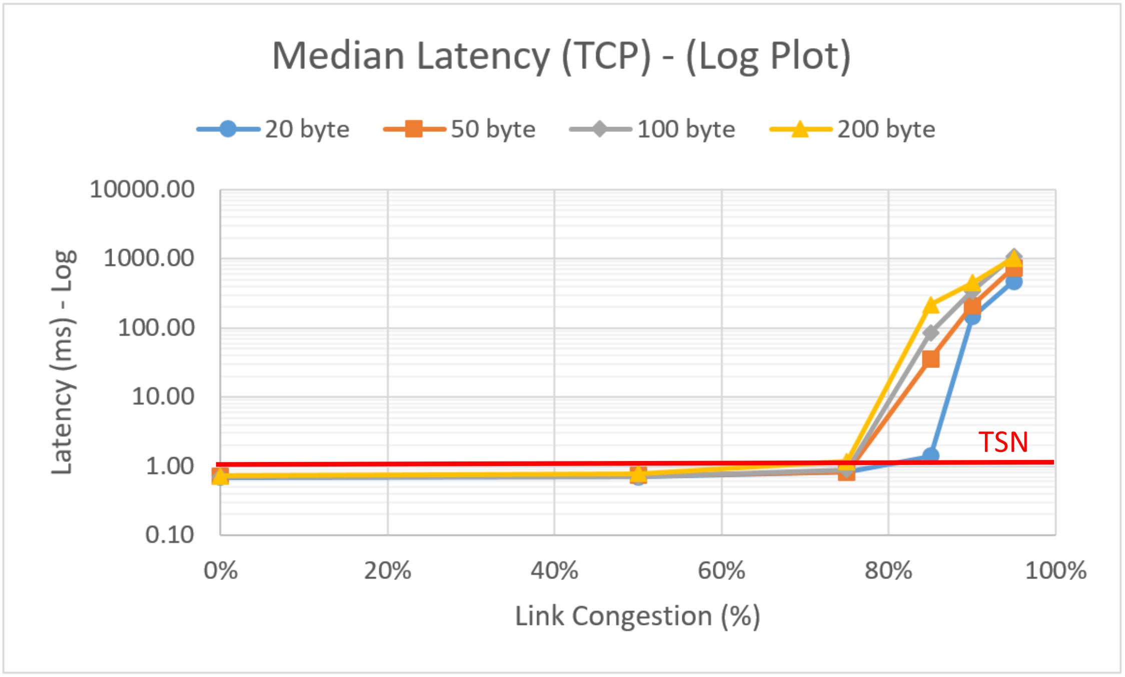


Figure 4.17: Median latency vs congestion for different TCP frame sizes – log plot.

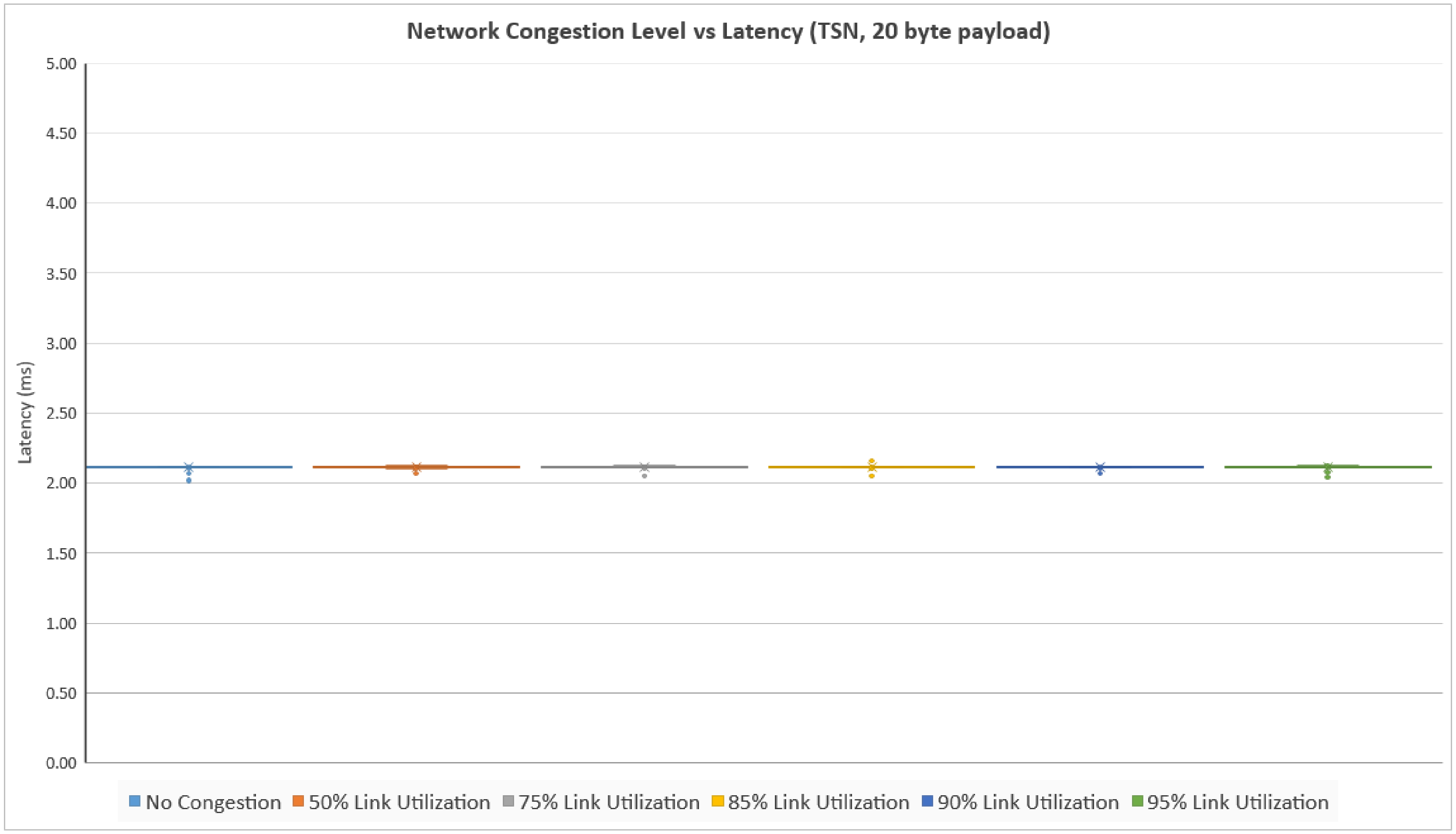


Figure 4.18: Distribution of network latency for a 20-byte TSN frame at different levels of network congestion.

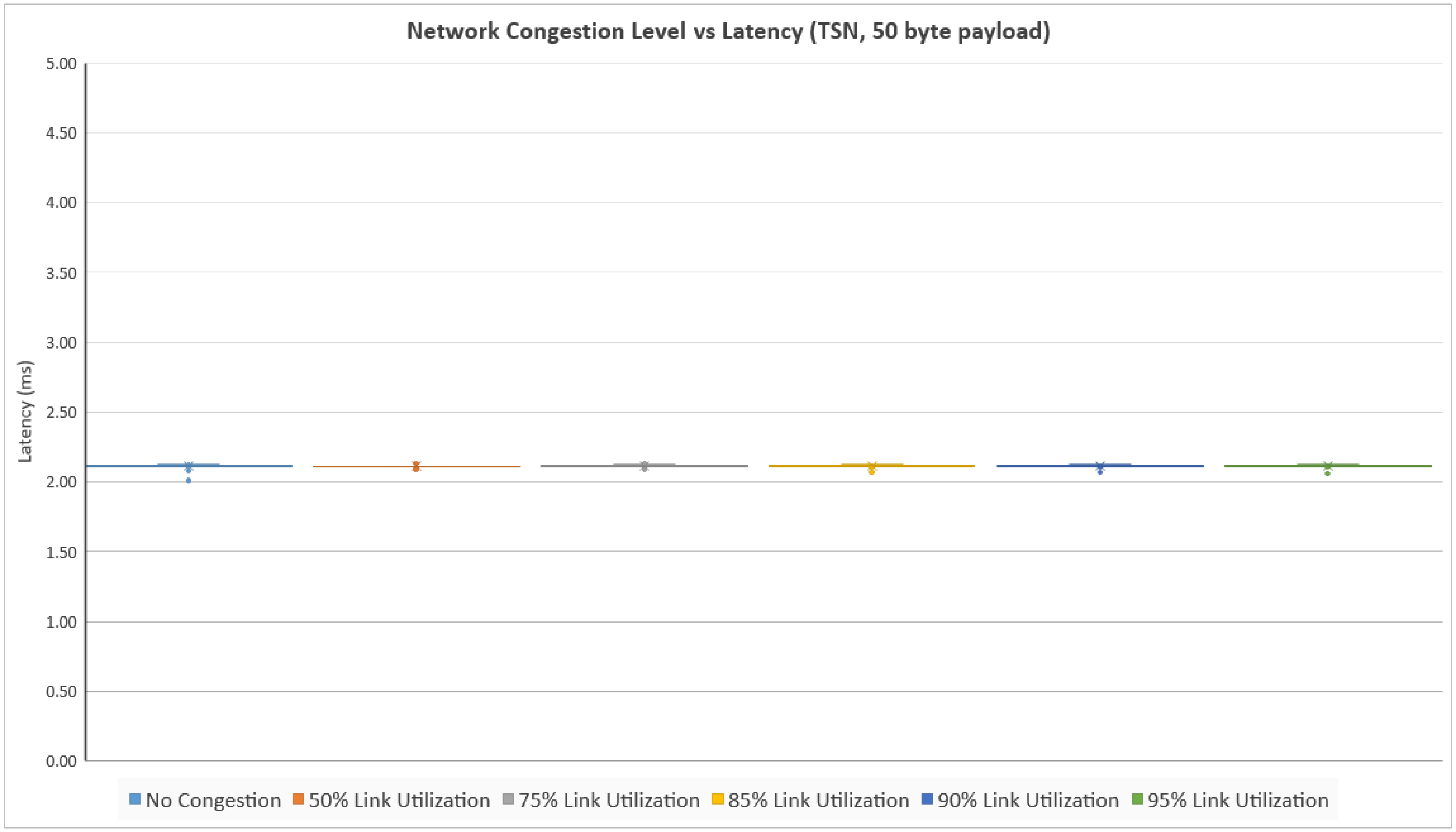


Figure 4.19: Distribution of network latency for a 50-byte TSN frame at different levels of network congestion.

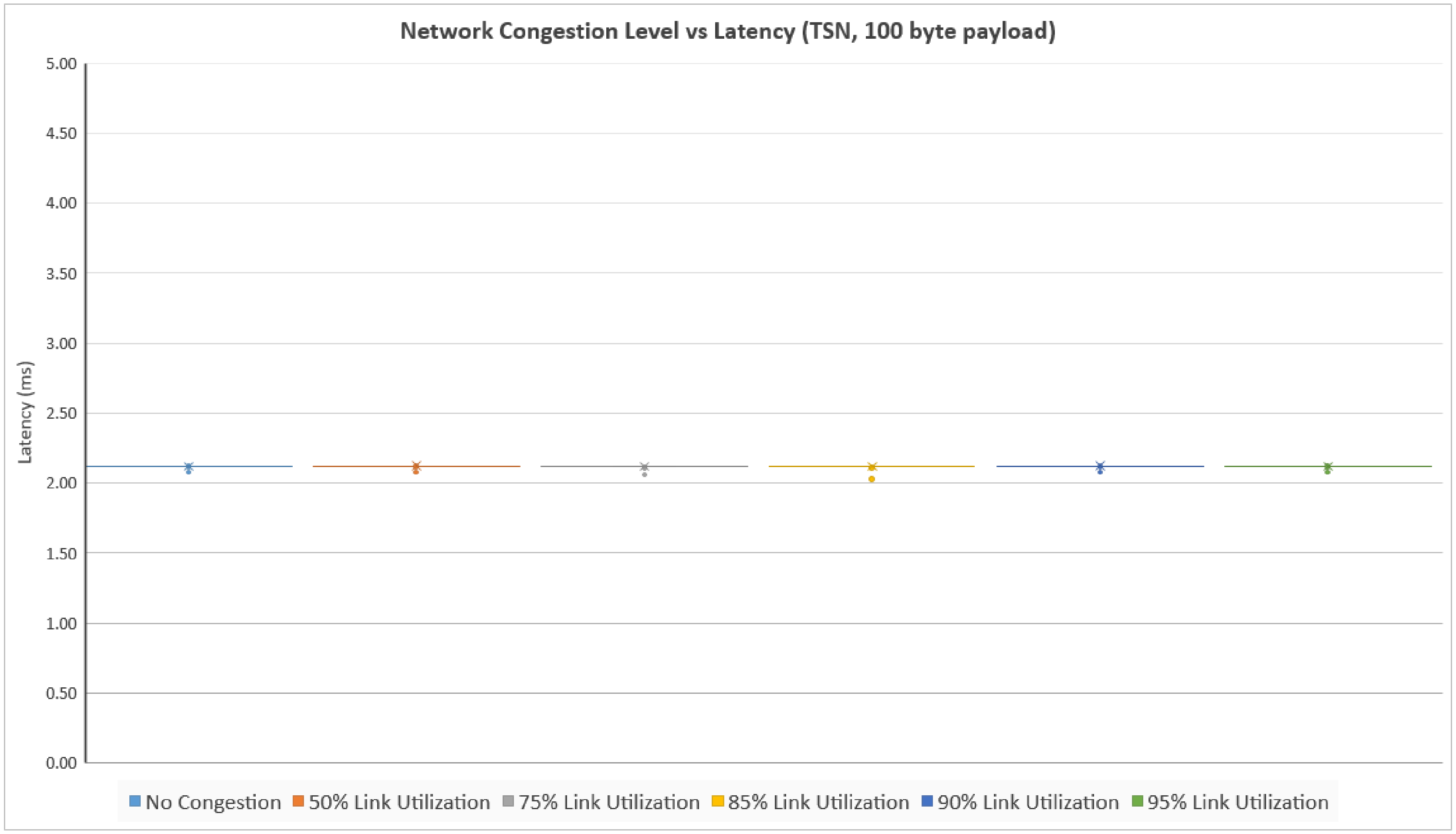


Figure 4.20: Distribution of network latency for a 100-byte TSN frame at different levels of network congestion.

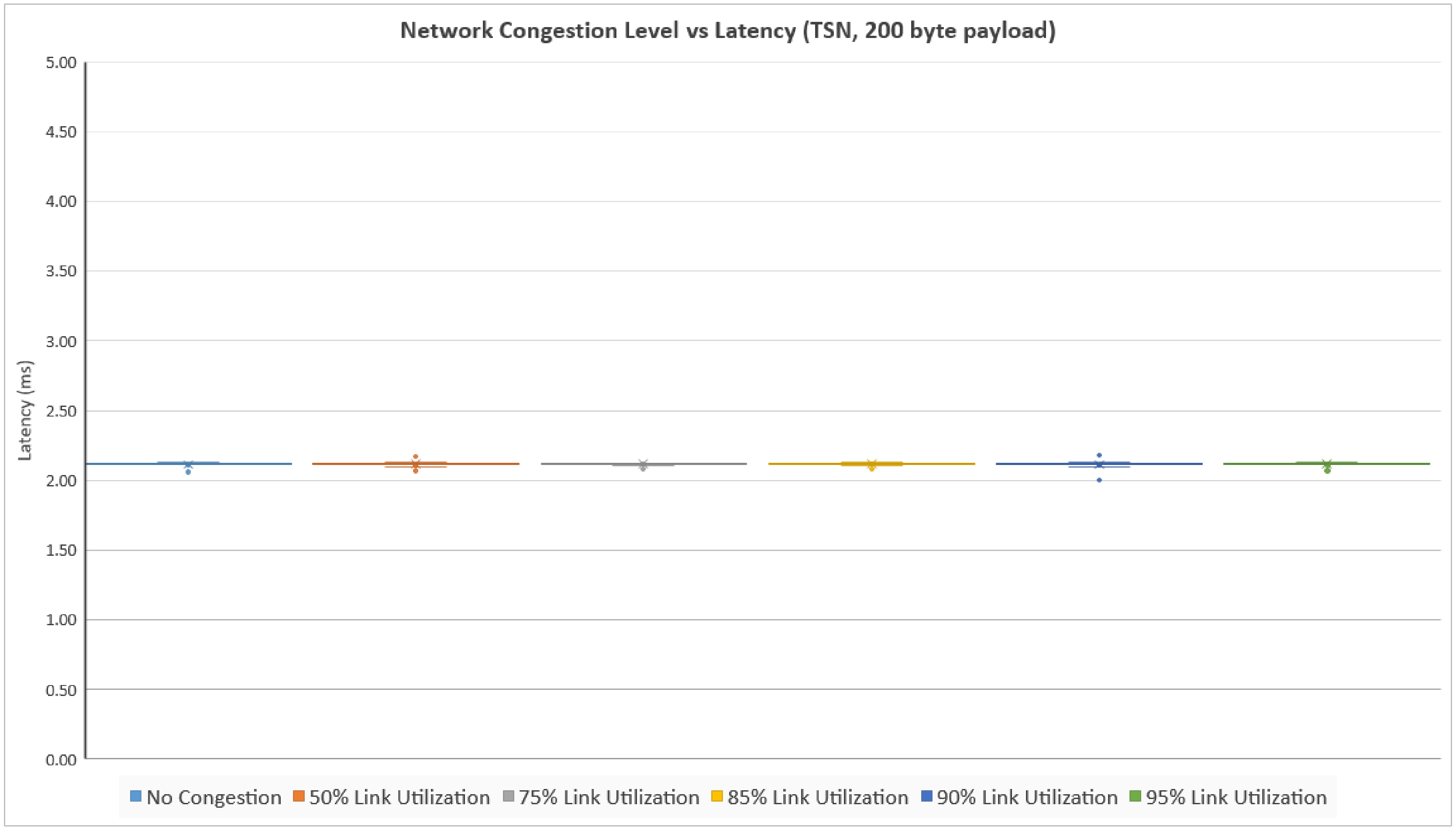


Figure 4.21: Distribution of network latency for a 200-byte TSN frame at different levels of network congestion.

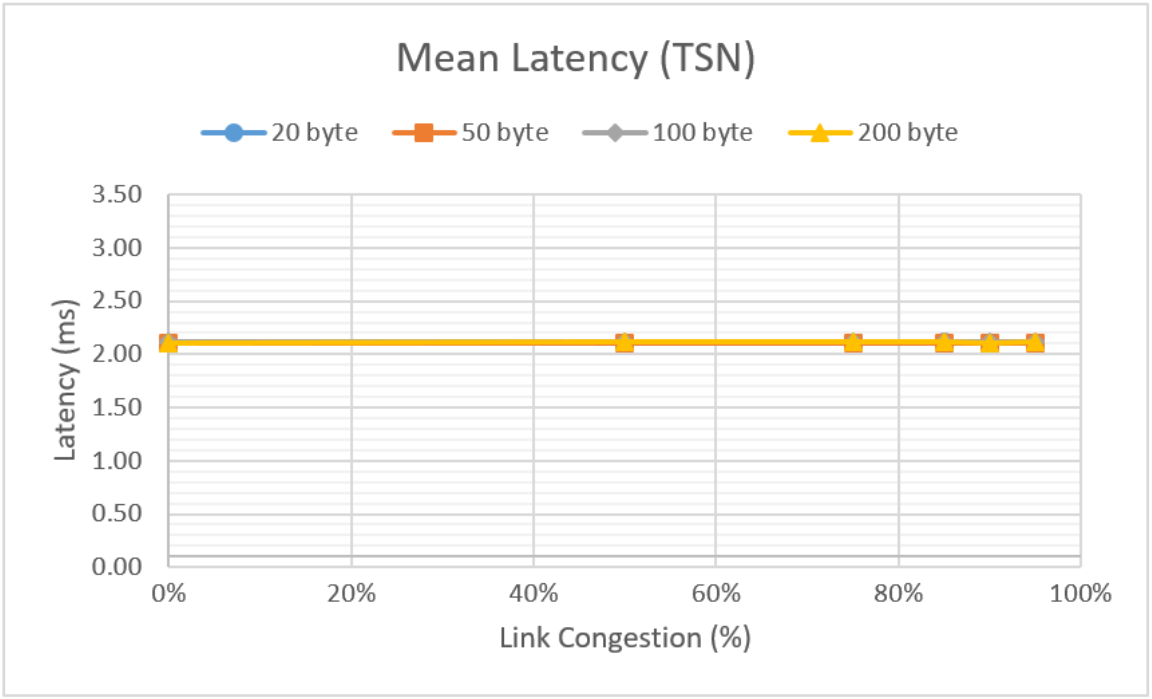


Figure 4.22: Average network latency vs congestion for different TSN frame sizes.

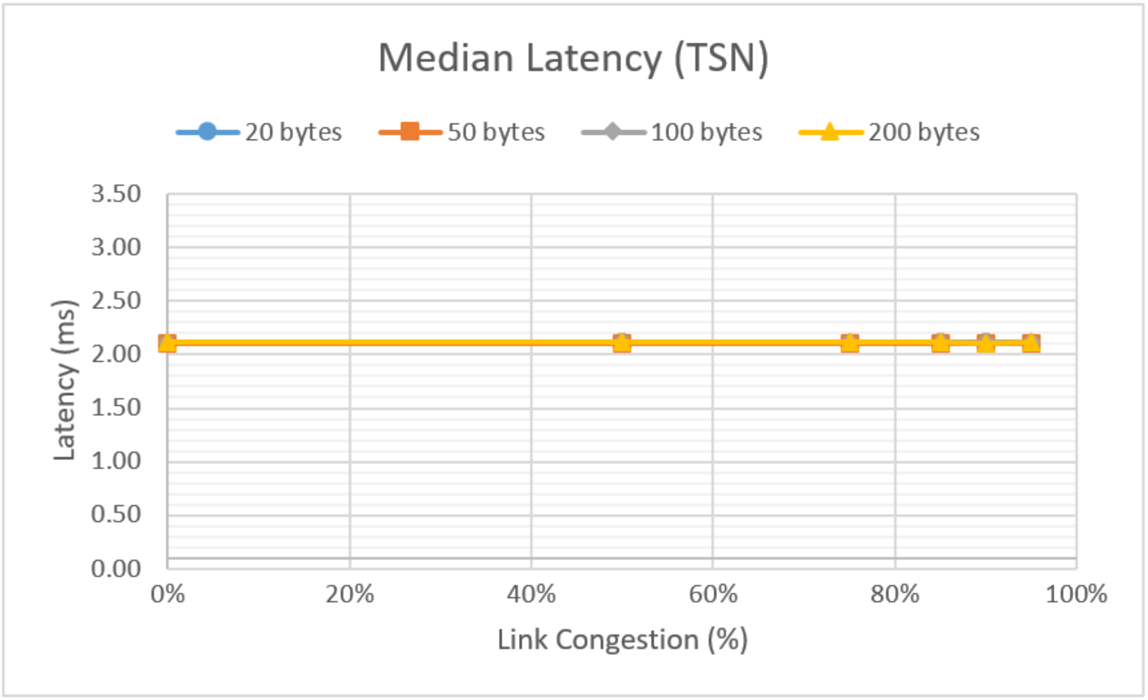


Figure 4.23: Median network latency vs congestion for different TSN frame sizes.

4.4 TSN Sync Waveform and Frequency Control

To compare the synchronization capabilities of using TSN as both the means for timing and for rapid dissemination of control signals, UDP and TSN were used to transmit time-varying frequency setpoints from the talker cRIO to the listener cRIO. The results of this test are shown below in Table 4.1. In this test, a connection was established and synchronized. An oscilloscope was then used to gather statistical data about the phase difference between the output signals from the cRIO talker and the cRIO listener. The signals were left to run for 10 minutes. Minimum (0%) and maximum (95%) network congestion was tested, as well as a static signal and a time varying signal. For these tests, the *Response Time Setpoint* was set to 15 ms, or just under a single 60 Hz cycle. Signal 1 (the top signal) in all tests represents the talker cRIO, while Signal 2 (the bottom signal) represents the signal tracking the setpoint given by the talker.

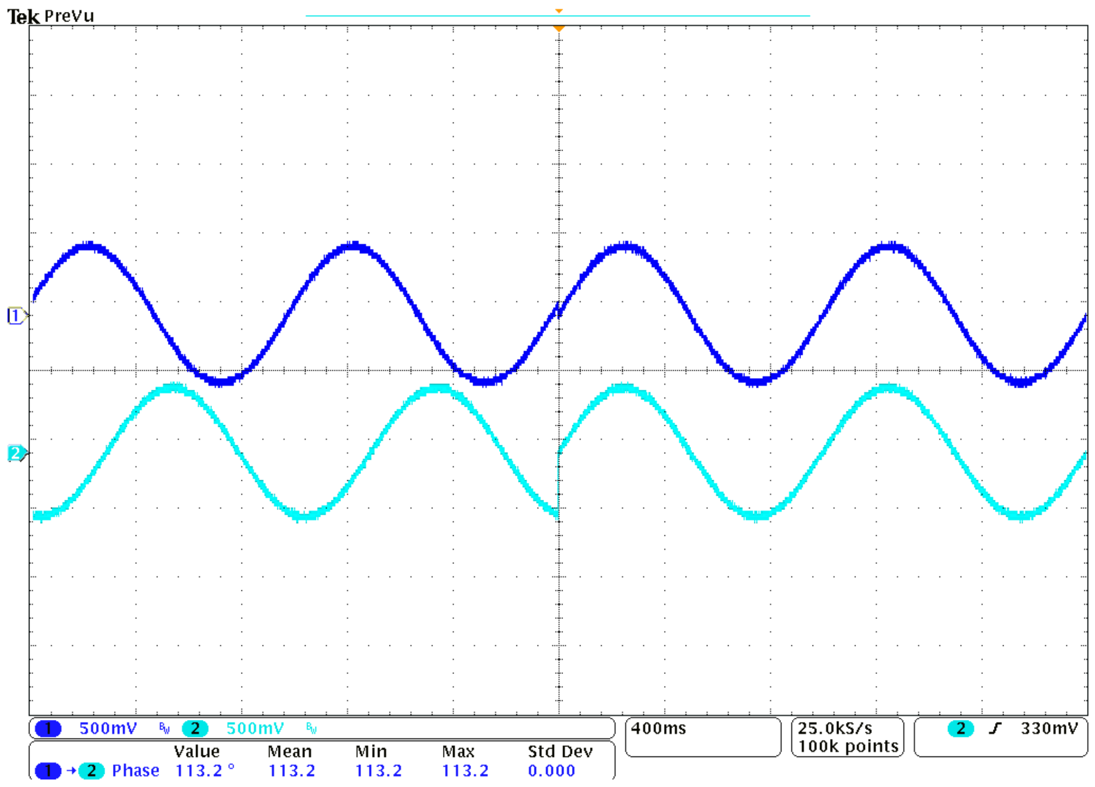
Due to the way the synchronization function implemented on the cRIO FPGA is designed, it was expected that all four results for the static case would be roughly equivalent. This is because, after the two signals have been synchronized for the first time, the signals no longer require communication (other than the built in 802.1AS time synchronization, which is maintained automatically) to maintain synchronization. This is what was observed in the tests shown in Table 4.1. During the initial synchronization of the signals, a new *Start Time* value is communicated from the first cRIO to the other.

Table 4.1: Synchronization statistics between UDP/TSN for static and time-varying signals at minimum and maximum network congestion.

Elapsed Time: 10 minutes		UDP		TSN	
		0%	95%	0%	95%
Static Frequency (60Hz)	Max Phase Diff	*2.48*	*2.78*	*2.74*	*2.703*
	Min Phase Diff	*-1.51*	*-1.10*	*-1.19*	*-0.97*
	Mean Phase Diff	*0.49*	*0.74*	*0.76*	*0.74*
	Std Dev Phase Diff	*0.56*	*0.55*	*0.59*	*0.50*
Time-Varying Frequency (59.5Hz – 60.5Hz)	Max Phase Diff	*3.53*	*48.91*	*2.92*	*2.83*
	Min Phase Diff	*-0.98*	*-23.05*	*-0.98*	*-1.61*
	Mean Phase Diff	*1.13*	*3.00*	*0.98*	*0.68*
	Std Dev Phase Diff	*0.71*	*21.24*	*0.62*	*0.61*

This value is used as the start point where the two signals will begin outputting a synchronized sine wave. If, after the first synchronization (shown in Figure 4.24) the frequency does not change, no further critical communication exchange takes place.

In the time-varying frequency case, the setpoint frequency is continuously changing. A loop generates a random value between -0.5 and 0.5 to either add to or subtract from the previous frequency setpoint. To avoid excessively large jumps, only a value less than 0.1 away from the previous change will be added/subtracted, preventing the frequency from changing too much too quickly (as can be seen in Figure 3.23). In the time-varying frequency case, it was anticipated that both streams at low congestion would continue to keep good synchronization, as the latency tests from Section 4.1 showed that UDP can perform as well or better than TSN in a network with little additional congestion. However, the high congestion case shows that UDP



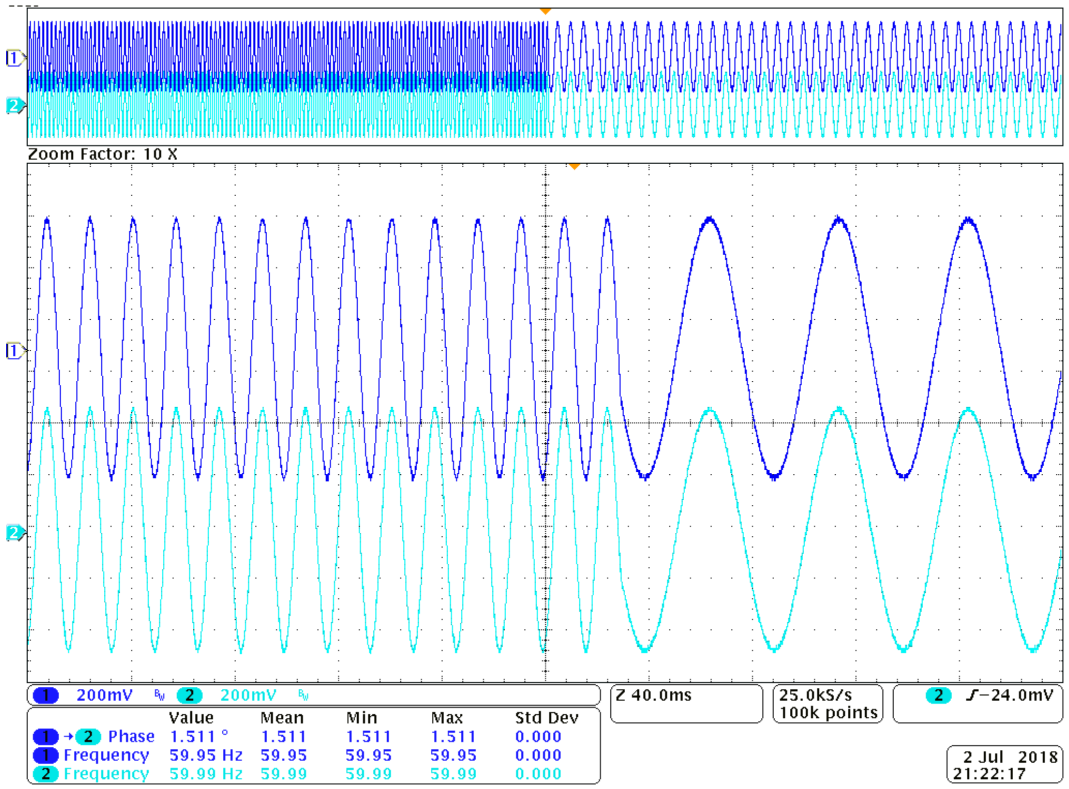
	Value	Mean	Min	Max	Std Dev
1 → 2 Phase	113.2 °	113.2	113.2	113.2	0.000

Figure 4.24: Initial synchronization of waveforms using TSN communication and 802.1AS timing.

synchronization quality is poor at high network utilization. Conversely, TSN continues to hold roughly the same level of synchronization regardless of the congestion level.

As is evident from Table 4.1, the average and standard deviation of the TSN signals show that a TSN communication assisted synchronization method is feasible, and can achieve a synchronization precision within approximately one degree of phase shift. There appears to be a slight positive phase bias between signals, in that the mean phase shift is slightly positive in each trial, and approaches 0.7 degrees. With some improvements to correct for this error, this value could be reduced to approach zero.

Figure 4.25 demonstrates a frequency step change from 60 Hz to 20 Hz. The two signals keep good synchronization even with a rapid frequency change. While such a drastic frequency change is not terribly realistic in a power system context, the large change helps demonstrate the speed and precision that the two signals are able to adjust output frequency at the same moment.



	Value	Mean	Min	Max	Std Dev
1 → 2 Phase	1.511 °	1.511	1.511	1.511	0.000
1 Frequency	59.95 Hz	59.95	59.95	59.95	0.000
2 Frequency	59.99 Hz	59.99	59.99	59.99	0.000

Figure 4.25: Synchronized frequency setpoint transition from 60 Hz to 20 Hz.

Figure 4.26 shows a time-varying frequency maintaining synchronization over a TSN network experiencing full network congestion. The base frequency shown is only 1 Hz. This was chosen to allow a greater degree of frequency shift to be captured in the short window of the oscilloscope screen. The true capabilities of the synchronization quality of using TSN to update the frequency setpoint in real-time are difficult to appreciate with static images in a document. Viewing the transitions live, the follower cRIO keeps near perfect synchronization with the master cRIO, demonstrating the potential of combining both 802.1AS time synchronization and 802.1Qbv latency guarantees to allow real-time sub-cycle frequency control in a microgrid system. Under the same circumstances, TCP and UDP (under higher network load) quickly lose synchronization, and phase shift grows to double-digit values within several minutes. Figure 4.27 and Figure 4.28 are captures of a UDP based session under full network congestion quickly losing synchronization at 60 seconds and 120 seconds. Figure 4.29 shows the same scenario 10 minutes after initial synchronization, using TSN as the communication medium.

4.5 Chapter Summary

The results of the latency experiments demonstrate that IEEE TSN is able (as expected) to maintain a given latency requirement regardless of data size of network conditions. While the results of this project's experiments measured TSN latency as 2 ms, lower latencies (with corresponding higher computation load on the cRIOs) were observed with setting changes in the network CNC.

UDP, while still maintaining relatively low latency, never the less experienced larger latencies as network congestion levels rose. Frame size also increased latency for UDP at a given network congestion level. TCP, as a protocol with loss packet detection built in, experienced extreme latency (compared to TSN and UDP) at higher congestion levels. In addition, much like UDP, larger data frame sizes also increased latency. TSN seems to live up to the promise of the new standards, demonstrating low latency regardless of the frame size or congestion level.

In addition, the synchronization of distributed reference waveforms was demonstrated successfully using TSN a means to rapidly update the frequency setpoint each cycle. Synchronization within approximately a single degree of phase shift was measured, and could likely be improved with further optimization.

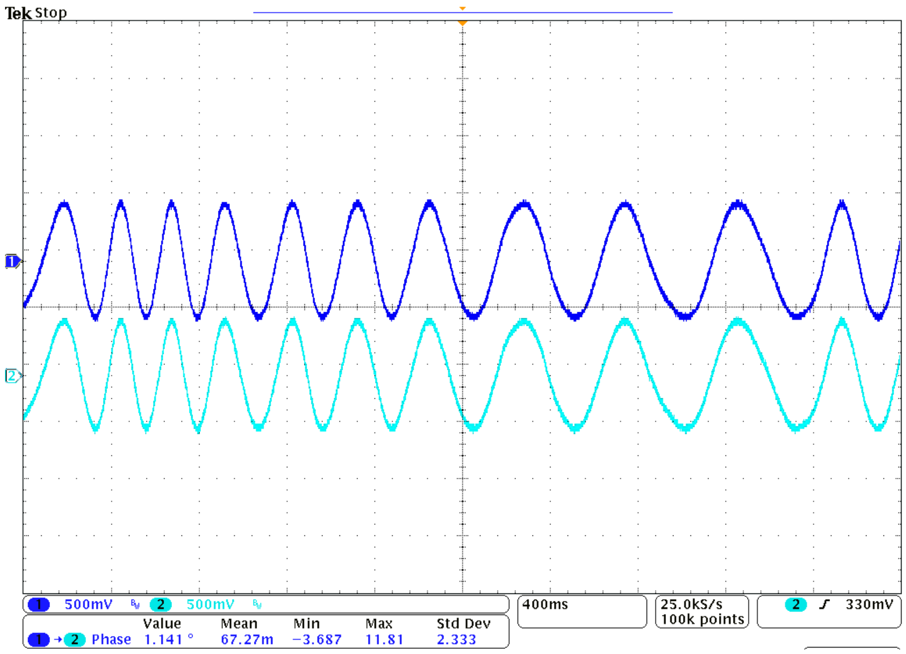

```
Tek Stop
1  500mV   2  500mV                     400ms     25.0kS/s      2  330mV
                                                  100k points
            Value     Mean     Min      Max      Std Dev
1→2 Phase   1.141 °   67.27m   −3.687   11.81    2.333
```

Figure 4.26: Synchronization of time-varying waveform using TSN under 95% network congestion.

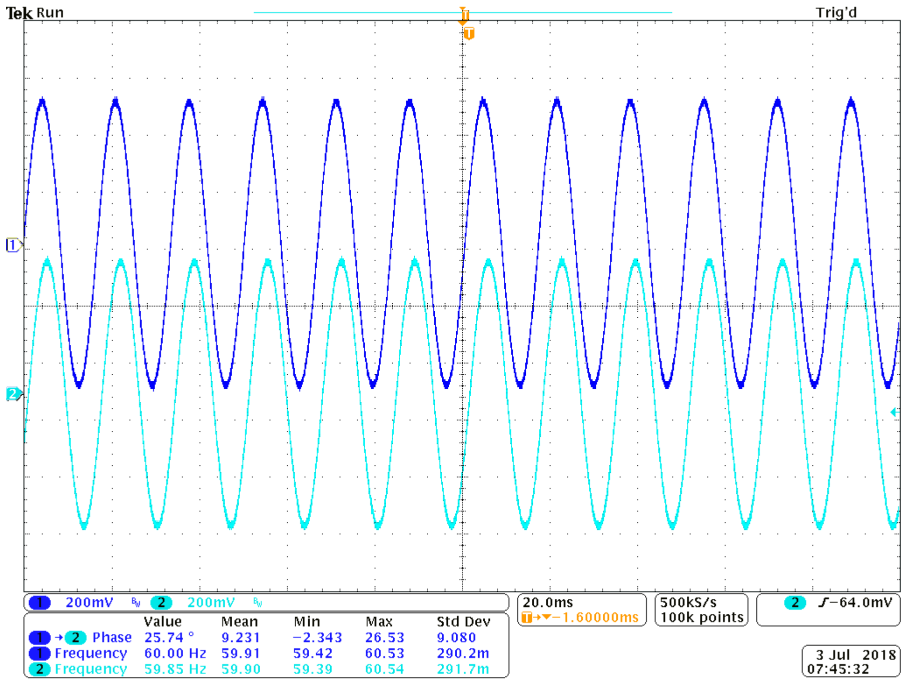
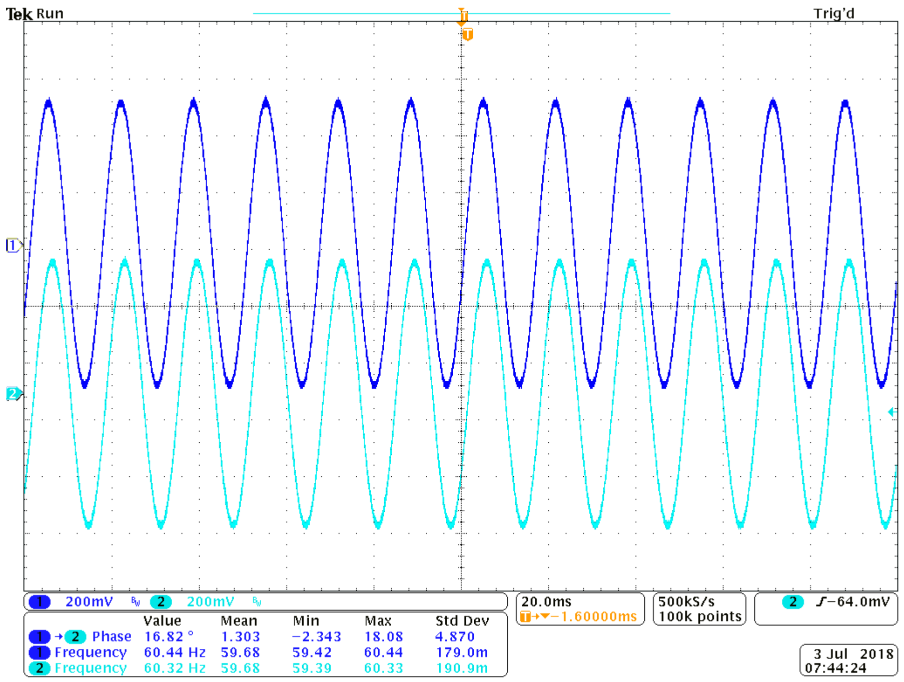
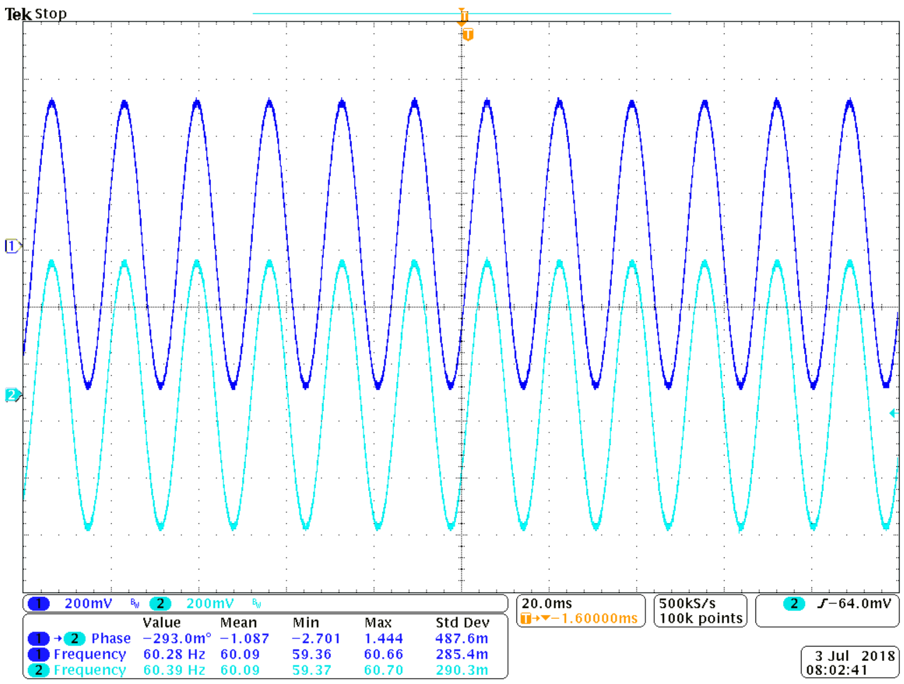

```
Tek Run                                                          Trig'd
1  200mV   2  200mV                     20.0ms         500kS/s      2  −64.0mV
                                        T→▼−1.60000ms  100k points
                Value      Mean    Min      Max      Std Dev
1→2 Phase       25.74 °    9.231   −2.343   26.53    9.080
1 Frequency     60.00 Hz   59.91   59.42    60.53    290.2m
2 Frequency     59.85 Hz   59.90   59.39    60.54    291.7m
                                                            3 Jul  2018
                                                            07:45:32
```

Figure 4.27: UDP synchronization of a time-varying frequency on a 95% congested network after 120 seconds. Max phase shift of 26 degrees observed.



```
1  200mV   2  200mV                 20.0ms          500kS/s      2  ʃ −64.0mV
                                    T→▼−1.60000ms   100k points
              Value     Mean    Min      Max     Std Dev
1 → 2 Phase   16.82 °   1.303   −2.343   18.08   4.870
1 Frequency   60.44 Hz  59.68   59.42    60.44   179.0m
2 Frequency   60.32 Hz  59.68   59.39    60.33   190.9m
                                                              3 Jul  2018
                                                              07:44:24
```

Figure 4.28: UDP synchronization of a time-varying frequency on a 95% congested network after 60 seconds. Max phase shift of 18 degrees observed.



```
1  200mV   2  200mV                 20.0ms          500kS/s      2  ʃ −64.0mV
                                    T→▼−1.60000ms   100k points
              Value      Mean     Min      Max     Std Dev
1 → 2 Phase   −293.0m°   −1.087   −2.701   1.444   487.6m
1 Frequency   60.28 Hz   60.09    59.36    60.66   285.4m
2 Frequency   60.39 Hz   60.09    59.37    60.70   290.3m
                                                              3 Jul  2018
                                                              08:02:41
```

Figure 4.29: TSN synchronization of a time-varying frequency on a 95% congested network after 10 minutes. Synchronization comparable to non-varying, low congestion tests.

5 Conclusion and Recommendations

As more distributed generation resources continue to come online, new methods of energy management and control are required to effectively integrate this influx of inconsistent generation into the existing grid while maintaining reliability. In most cases, the presence of these distributed resources can lead to increased reliability and reduced costs when leveraged effectively. Smart microgrids are an effective and innovative method of aiding in the integration of these distributed resources, as well as leveraging these resources to reduce dependence on the main grid and energy costs. In order to optimize and control the wide area administered by a microgrid (or smart grid), a centralized controller needs full awareness of the state of the microgrid at every moment, as well as the ability to quickly alter the operating state of the microgrid at a moment's notice. Effective communication is a core requirement of any microgrid or smart grid implementation that hopes to maximize economic and performance considerations in this manner.

IEEE Time Sensitive Networking technology has the potential to transform how power system communication networks are interfaced together. The ability to utilize a high speed Ethernet based physical backbone, while also maintaining the latency, synchronization, and reliability features of previously proprietary or application specific solutions will lead to novel methods of communication and control in a variety of applications within the grid area.

Over the course of this exploration of IEEE Time Sensitive Networking's potential for widespread use in microgrid systems, the timing and latency guarantee features of the standards were the main focus. The timing features of TSN allow for improved time synchronization over the same physical medium as control and monitoring communication, obviating the need for additional hardware such as GPS receivers, while providing higher precision than other methods such as NTP. In addition, providing every node on a network with an automatic synchronized time reference can allow for the relatively simple implementation of tightly scheduled synchronized actions, to allow for superior microgrid coordination performance.

The latency guarantee features of TSN show equal promise, demonstrating the ability to maintain a consistent and minimal level of latency, despite heavy network impairment and variable payload size. The addition of this feature to a microgrid system would allow for the rapid dissemination of control signals to distributed entities using the simple and expandable Ethernet medium for the physical backbone. In addition to latency and synchronization features, the additional TSN features not covered in depth promise to aid in improving communication reliability and control through the addition of path redundancy and filtering policies. The fact that these standards have all been developed under the aegis of the 802.1 Local Area Network (LAN) standard further allows the features of TSN to be utilized with existing physical infrastructure, while allowing communication backwards compatibility with non-TSN enabled hardware.

To facilitate testing of the TSN 802.1AS and 802.1Qbv standards, a test network composed of two Cisco TSN switches, two NI Compact RIOs, and two virtual machines to heavily load the network was assembled. By utilizing the iPerf network bandwidth measurement utility, the network could be loaded to any degree required to create repeatable tests of network impairment. Two other protocols, UDP and TCP, were chosen to act as the facilitating protocols to test and compare the capabilities of a non-TSN link-layer.

The results of these tests demonstrated that it was indeed possible to maintain minimal latency on a highly loaded network using TSN. This was not shown to be possible with UDP or TCP. In addition, the synchronized timing features present in TSN were demonstrated to be useful for generating synchronized reference waveforms for use in a microgrid setting for inverter synchronization.

However, despite its promising results in experimental tests compared with UDP and TPC, TSN is not yet ready for widespread adoption. The availability of commercial hardware that supports the standards is extremely slim, and software for administering TSN networks also is immature and lacking functionality. The suite of standards has the potential to become extremely prolific in industrial applications, but more time is needed for that option to be truly viable.

In the context of the work performed in support of this thesis, more research is needed to determine if the results presented here scale appropriately with an increase in network complexity and size. Though the results of the test were clear, the limitations of testing with two switches are apparent. To further test the abilities of TSN compared to UDP or TCP, tests within a larger test network are required. In addition, testing faster latency requirements and larger data payloads is another area where the results of this work could be extended. While the results of the experiments conducted to characterize the potential limits of TSN are interesting from an academic perspective, future work should be explored pushing these limits to augment other real world applications.

BIBLIOGRAPHY

[1] R. Margolis, D. Feldman, and D. Boff, "Q4 2016/Q1 2017 Solar Industry Update," National Renewable Energy Lab (NREL), Golden, CO (United States), 2017.

[2] G. Ditzel and P. Didier, "Time sensitive network (tsn) protocols and use in ethernet/ip systems," in *2015 ODVA Industry Conference & 17th Annual Meeting*, 2015.

[3] H. Laine, "Simulating Time-Sensitive Networking," *Technical University of Denmark, Lyngby, Denmark*, 2015.

[4] K. Sridharan, K. Goossens, N. Concer, and H. B. Vermeulen, "Investigation of Time-Synchronization over Ethernet In-Vehicle Networks for automotive applications," Master Thesis. Eindhoven: Eindhoven University of Technology, 2015.

[5] M. Wlas, M. Gackowski, and W. Kolbusz, "The Ethernet POWERLINK Protocol for smart grids elements integration," in IEEE *International Symposium on Industrial Electronics (ISIE)*, 2011, pp. 2070-2075.

[6] A. Bani-Ahmed, L. Weber, A. Nasiri, and H. Hosseini, "Microgrid communications: State of the art and future trends," in IEEE *International Conference on Renewable Energy Research and Application (ICRERA), 2014*, pp. 780-785.

[7] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of IEEE 802.1 AS in large-scale industrial automation networks," in IEEE *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017*, pp. 273-282.

[8] C. Cho, J.-H. Jeon, J.-Y. Kim, S. Kwon, K. Park, and S. Kim, "Active synchronizing control of a microgrid," *IEEE Transactions on Power Electronics*, vol. 26, no. 12, pp. 3707-3719, 2011.

[9] W. Gu, G. Lou, W. Tan, and X. Yuan, "A nonlinear state estimator-based decentralized secondary voltage control scheme for autonomous microgrids," *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4794-4804, 2017.

[10] "IEEE Approved Draft Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE P802.1Qcc/D2.3, May 2018*, pp. 1-214.

[11] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1-300.

[12] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine,* vol. 11, no. 1, pp. 17-27, 2017.

[13] C. Schwaegerl and L. Tao, "The microgrids concept," *Microgrids*, pp. 1-24, 2013.

[14] N. A. Luu, "Control and management strategies for a microgrid," Université de Grenoble, 2014.

[15] M. S. Golsorkhi, D. Lu, Q. Shafiee, and J. M. Guerrero, "Distributed voltage control and load sharing for inverter-interfaced microdrid with resistive lines," in IEEE *Energy Conversion Congress and Exposition (ECCE), 2016,* pp. 1-7.

[16] M. Castilla, A. Camacho, P. Martí, M. Velasco, and M. M. Ghahderijani, "Impact of clock drifts on communication-free secondary control schemes for inverter-based islanded microgrids," *IEEE Transactions on Industrial Electronics,* vol. 65, no. 6, pp. 4739-4749, 2018.

[17] S. Zuo, A. Davoudi, Y. Song, and F. L. Lewis, "Distributed finite-time voltage and frequency restoration in islanded AC microgrids," *IEEE Transactions on Industrial Electronics,* vol. 63, no. 10, pp. 5988-5997, 2016.

[18] A. Dimeas, A. Tsikalakis, G. Kariniotakis, and G. Korres, "Microgrids control issues," *Microgrids,* pp. 25-80, 2013.

[19] Q. Shafiee, J. M. Guerrero, and J. C. Vasquez, "Distributed secondary control for islanded microgrids—A novel approach," *IEEE Transactions on Power Electronics,* vol. 29, no. 2, pp. 1018-1031, 2014.

[20] Cisco Systems, "Time-Sensitive Networking: A Technical Introduction," Whitepaper 2017.

[21] "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," *IEEE Std 802.1AS-2011*, pp. 1-292.

[22] T. Kovácsházy, "Towards a quantization based accuracy and precision characterization of packet-based time synchronization," in IEEE *International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), 2016,* pp. 1-6.

[23] H.-T. Lim, D. Herrscher, L. Völker, and M. J. Waltl, "IEEE 802.1 AS time synchronization in a switched Ethernet based in-car network," in IEEE *Vehicular Networking Conference (VNC), 2011*, pp. 147-154.

[24] "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015 (Amendment to Std 802.1Q--- as amended by Std 802.1Qca-2015, Std 802.1Qcd-2015, and Std 802.1Q---/Cor 1-2015),* pp. 1-57, 2016.

[25] N. L. Díaz, A. C. Luna, J. C. Vasquez, and J. M. Guerrero, "Centralized control architecture for coordination of distributed renewable generation and energy storage in islanded ac microgrids," *IEEE Transactions on Power Electronics,* vol. 32, no. 7, pp. 5202-5213, 2017.

[26] I. Penn, "California Will Require Solar Power for New Homes," in *The New York Times*, ed, May 9, 2018, p. B1.

[27] J. Lai, H. Zhou, W. Hu, X. Lu, and L. Zhong, "Synchronization of hybrid microgrids with communication latency," *Mathematical Problems in Engineering,* vol. 2015, 2015.

[28] M. S. Golsorkhi and D. D. Lu, "A control method for inverter-based islanded microgrids based on VI droop characteristics," *IEEE Transactions on Power Delivery*, vol. 30, no. 3, pp. 1196-1204, 2015.

[29] G. Lou, W. Gu, L. Wang, B. Xu, M. Wu, and W. Sheng, "Decentralised secondary voltage and frequency control scheme for islanded microgrid based on adaptive state estimator," *IET Generation, Transmission & Distribution,* vol. 11, no. 15, pp. 3683-3693, 2017.

[30] S. S. Thale and V. Agarwal, "Controller area network assisted grid synchronization of a microgrid with renewable energy sources and storage," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1442-1452, 2016.

[31] C. Jamroen, N. Kesorn, A. Pichetjamroen, and S. Dechanupaprittha, "Impact of communication delays on PEVs charging power control for frequency stabilization in remote microgrid," in IEEE *Asia-Pacific Power and Energy Engineering Conference (APPEEC), 2017*, pp. 1-6.

[32] R. Hao, Z. Jiang, Q. Ai, Z. Yu, and Y. Zhu, "Hierarchical optimisation strategy in microgrid based on the consensus of multi-agent system," *IET Generation, Transmission & Distribution*, vol. 12, no. 10, pp. 2444-2451, 2018.

[33] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on smart grid communication infrastructures: Motivations, requirements and challenges," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 5-20, 2013.

[34] A. Aggarwal, S. Kunta, and P. K. Verma, "A proposed communications infrastructure for the smart grid," in IEEE *Innovative Smart Grid Technologies (ISGT), 2010*, pp. 1-5.

[35] M. Haripriya and C. Vasanthanayaki, "CAN based grid synchronisation technique of a micro grid with Renewable source," in IEEE *International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT),* 2017, pp. 1-4.

[36] T. Degner, N. Soultanis, A. Engler, and A. G. d. Muro, "Intelligent local controllers," *Microgrids*, pp. 81-116, 2013.

[37] J. Schiffer, C. A. Hans, T. Kral, R. Ortega, and J. Raisch, "Modeling, Analysis, and Experimental Validation of Clock Drift Effects in Low-Inertia Power Systems," *IEEE Transactions on Industrial Electronics,* vol. 64, no. 7, pp. 5942-5951, 2017.

[38] S. Adhikari and F. Li, "Coordinated Vf and PQ control of solar photovoltaic generators with MPPT and battery storage in microgrids," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1270-1281, 2014.

[39] J. Li, J. Su, X. Yang, and T. Zhao, "Study on microgrid operation control and black start," in IEEE *4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT),* 2011, pp. 1652-1655.

[40] J. Pacner, O. Ryavy, and M. Veda, "On the Evaluation of Clock Synchronization Methods for Networked Control Systems," in IEEE *3rd Eastern European Regional Conference on the Engineering of Computer Based Systems (ECBS-EERC),* 2013, pp. 161-162.

[41] Z. Zhang, S. Gong, A. D. Dimitrovski, and H. Li, "Time synchronization attack in smart grid: Impact and analysis," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 87-98, 2013.

[42] B. Snover and I. Knox, *Time Sensitive Networking: Time Synchronization and Deterministic Communication with cRIO*. Technical Presentation, NI Week, 2017.

[43] I. Patrao, R. González-Medina, S. Marzal, G. Garcerá, and E. Figueres, "Synchronization of power inverters in islanded microgrids using an FM-modulated signal," *IEEE Transactions on Smart Grid,* vol. 8, no. 1, pp. 503-510, 2017.

[44] H. M. Antunes, S. M. Silva, D. I. Brandao, R. V. Ferreira, and C. Braz de Jesus Filho, "Analysis of a grid-forming converter based on repetitive control in centralized AC microgrid," in *IEEE 8th International Symposium on Power Electronics for Distributed Generation Systems (PEDG),* 2017, pp. 1-8.

[45] K. Tan, X. Peng, P. L. So, Y. C. Chu, and M. Chen, "Centralized control for parallel operation of distributed generation inverters in microgrids," IEEE *Transactions on Smart Grid,* vol. 3, no. 4, pp. 1977-1987, 2012.

[46] M. M. Rana, L. Li, S. W. Su, and W. Xiang, "Microgrid State Estimation: A Distributed Approach," *IEEE Transactions on Industrial Informatics,* 2017.

[47] J. Imtiaz, J. Jasperneite, and L. Han, "A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication," in IEEE *Conference on Emerging Technologies & Factory Automation (ETFA), 2009*, pp. 1-8.

[48] W. Elamin and M. Shaaban, "Real-time centralized control scheme for energy management in smart microgrids," in IEEE *Power & Energy Society General Meeting*, 2017, pp. 1-5.

VITA

Montie Smith received his BS in Electrical Engineering from The University of Tennessee in December 2016. Since then, he has worked as a graduate research assistant as part of the Center for Ultra-Wide-Area Resilient Electric Energy Transmission Networks (CURENT). As a member of CURENT, Montie worked predominantly on communication systems for distributed controllers for a dynamic smart microgrid.