Demonstrating Redundancy Advantages of a Three-Channel Communication Protocol

Scott Culbreth, Scott Graham
Air Force Institute of Technology, Wright-Patterson Air Force Base, United States

Scott.Culbreth.1@us.af.mil
Scott.Graham@afit.edu

Abstract: Multi-Channel communications have the potential to provide advantages in security and redundancy. One widespread example of additional security is the use of 2 Factor Authentication wherein an authorization code is sent via a separate channel. As another example, spread spectrum technology offers resilience against channel interference. However, no currently deployed communication protocols take advantage of the full spectrum of security and performance gains that can be obtained through transmitting data over multiple channels.

Taking inspiration from Redundant Array of Inexpensive Disks (RAID) and their use of data striping and mirroring, a secure multi-channel communication protocol was developed that is able to have greater security than an equivalent single channel system while also having greater resiliency against data corruption, therefore requiring fewer, if any, retransmissions than a single channel system when operating in a low availability environment. This approach admittedly comes with significant overhead, both in the use of additional channels and the need for additional processing. Whether the security and availability gains are worth the costs is an open question, with specific answers highly dependent on the particular applications.

A specific multi-channel communication protocol, with three independent channels, and incorporating duplication on the bit level, was built and exercised within the OMNeT++ simulation environment in order to examine specific aspects of performance and security of the protocol.

This exercise demonstrates that a secure multi-channel protocol operates with less latency than an equivalent single channel system when experiencing less than 50% channel corruption due to adversarial injection, resulting in reduced data loss and need for re-transmissions.

Keywords: Multi-Channel, Security, Encryption, Networking

1. Introduction

As threats to computer security evolve, our methods of combating security threats must also evolve. Direct threats to the confidentiality, integrity, and availability of communications are developed daily; this research seeks to develop an effective security-focused communication protocol operating at the session layer, taking advantage of diverse multi-channel communications. The proof of concept for this protocol includes a three-channel system that sends individual bits of data over two separate channels, which is referred to as a duplication factor of two (Simon 2021). The implementation incorporates bit scrambling, error detection and error correction in an attempt to address confidentiality, integrity, and availability of the system. The protocol was implemented and tested in simulation to demonstrate its effectiveness, and test the limitations. Simulation demonstrated improved performance regarding confidentiality, integrity, and availability in non-permissive environments without using encryption.

1.1 Background

Computer security is often represented using the Confidentiality, Integrity, and Availability (CIA) triad. These three factors often guide the actions taken to show that a computer network is secure and robust. Threats to confidentiality are those that allow an adversary to glean unintended information from what was intended to be a secret communication. This is often the primary area considered in “security” as it is the protection of secret information. Confidentiality threats include decrypting secret messages, but data leakage is also a major concern. Integrity ensures that the message received is the message that was sent, i.e., that data is not lost or manipulated in transit, either through random noise within the transmission medium, or through adversaries suppressing or injecting information. Finally, availability is a key consideration for security. If a service is unavailable, it does not matter if it is unreadable to an adversary as it is unusable by the intended parties, thus a system must be available to be viable.

1.2 Problem Statement

The desire for computer communication protocols to be fast, secure, reliable and inexpensive is obvious, but these objectives present a trade-off. Often a fast and secure system is not reliable and affordable while a fast and affordable system is generally less secure or reliable. A single channel system that is fast, secure, and reliable is unlikely to be affordable. This research tackles this issue with a security focused multi-channel protocol. The protocol is focused on data leakage that could occur across the channels, or data an eavesdropper could gain by listening in on the channel and sniffing packets. In a single channel setup, all the data is available to a potential eavesdropper and all security would have to be based on cryptographic means. Encryption algorithms can carry significant overhead and can in some cases be unrealistic to use. Considerations for potential data injection attacks should be made as well. Finally, although networks typically offer multiple paths to a destination, most routing only takes advantage of a single path.

1.3 Hypothesis

A secure multi-channel protocol with a duplication factor of two can provide higher availability, with less data leakage, over an equivalent single channel system. Additionally, under heavy levels of corruption, a redundant multi-channel communication system has the ability to get messages through in less total time or with less latency per message/packet.

2. Related Work

This section provides a summary of existing networking technologies, how they inspired the creation of the protocol, and tests of the networking protocol. These include routing, data fragmentation and scrambling, as well as general network security requirements and background. Additionally, the simulation environment will be discussed, providing context on how the tests and exercises were conducted.

2.1 Multi-Channel Use

The use of multiple channels has existed in various forms for decades. The Public Switched Telephone Network adopted Signalling System 7 (SS7) in 1989, with both a communication channel, as well as an out of band signalling and control channel to operate and administer the network. This allowed for call data to operate on one channel while not being in competition with necessary signals to operate the network (Modarressi 1990).

Redundant Array of Inexpensive Disk (RAID) arrays can also be considered a multi-channel network with each disk acting as a channel. The methods used to duplicate and stripe data for redundancy, efficiency, and performance can be applied to other multi-channel uses (Hughes 2005).

Finally, Two-Factor Authentication (2FA), is another example of multi-channel use, and in this case, specifically for security. 2FA uses multiple channels to confirm the identity of a person attempting to access a system or account, beyond the traditional username and password, by sending an identifying code through a separate channel. This extra requirement helps ensure that the person providing login credentials has authorized access, as compromising both the username and password, as well as the secondary channel is significantly difficult (Dmitrienko 2014). Typically, this confirmation comes in the form of a text message or call to a mobile device, but various schemes exist. For example, the use of a secondary smart card provided in combination with login credentials or biometrics can be considered 2FA. Overall, the use of multiple channels creates significant obstacles for any adversary to overcome and results in significantly greater security (Dmitrienko 2014).

2.2 Multipath Transmission Control Protocol (MPTCP)

MPTCP is an extension of the traditional Transmission Control Protocol (TCP), utilizing multiple paths to improve performance and redundancy. Traditional TCP is limited to communication between two separate addresses, or interfaces, on a single channel. MPTCP extends TCP from communication between two interfaces to communication between two hosts or endpoints (Ford 2020). This enables multiple different interfaces from a single host to establish individual TCP connections and utilize both simultaneously. A typical example of this in practice would be a smart phone that has both WiFi and cellular connection capability. In a standard TCP connection, these two interfaces cannot be used in conjunction. However, when utilizing MPTCP, both the WiFi and cellular interfaces can be used simultaneously and thus allow for multiple paths to be used, allowing traffic to be sent over either of the two (Bonaventure 2016). If, for example, the user loses the WiFi connection, traffic

is sent over the cellular connection alone, transparent to the user. Overall, this creates a protocol that is capable of extending a widely adopted protocol in the form of TCP and using multiple paths to improve redundancy.

2.3 OMNeT++

OMNeT++ is "an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators." (What is OMNeT++? 2022). This framework allows the creation, testing, and development of network characteristics within a Graphical User Interface supported, C++ based, environment. OMNeT++ has countless pre-built modules from which simple networks or queues can be modelled, or complex networking relationships can be analysed. Features such as Peer-to-Peer networks, wired and wireless connectivity, and cloud-based solution modelling are available. Additionally, due to the C++ based nature of the environment, any specific modules for a particular scenario can be created and integrated with the existing models and packages. This creates a seamless and easy to work with simulation environment specifically tailored to the work being done.

3. Methodology

The intent of this proposed protocol is to increase security by utilizing multiple channels and bit scrambling while also increasing availability by overcoming packet loss and corruption/injection through the multi-channel setup. Additionally, performance should also improve over a single channel system. We begin by defining requirements for the protocol, then developing to those requirements, followed by identifying metrics to quantify the protocol's performance and finally exercising the protocol in simulation to determine performance and results.

3.1 Defining Requirements

Defining the requirements for the protocol is essential to guide the development and implementation of the protocol. For this implementation, the protocol will have three channels. As described in Section 2, multiple channels are not a new concept, however the spreading of data across multiple channels for security benefits is novel (Simon 2021).

The next requirement is that of a duplication factor of 2 (DF = 2). This means that each bit of data is sent over two of the three available channels. This allows for redundancy; as corruption or suppression of data occur, the data can still reach the destination over another channel.

An additional requirement is for the protocol to be able to detect corrupted packets due to random bit flips or from malicious action (injection). The protocol should also implement some form of data scrambling. Data scrambling is the process through which individual bits are taken from original data and, as the name implies, scrambles them into different locations within the packet. An eavesdropper who is not privy to the scrambling code would find the data unreadable. In addition, the bits are spread across channels, such that no channel has the full set of bits, which we call "data dropout", creating a significant challenge for eavesdroppers looking to extract usable data within a stream (Bao 2017). While not as secure as encryption, scrambling combined with data dropout significantly increases attacker difficulty as the target data is now out of order and incomplete.

3.2 Developing the Protocol

The proposed protocol assumes operation at the session layer. The first challenge of a multi-channel setup is creating the mechanisms which handle the decision making and processing of packets into and out of streams. The initial setup and protocol design developed in (Simon 2022) is followed in this new protocol.

The *Dissembler* and *Assembler* work in pairs at the session layer to send and receive the packets respectively. The *Dissembler* distributes the data from the application and creates the *Channel Packets* which contain the data sent along the respective channel, along with all necessary headers and footers for the protocol to operate. This operates similar to the MPTCP endpoints described within Section 2.2 where the (*Assembler Dissembler*) pair are the endpoints that create the multiple paths. The primary difference is the security elements implemented within the *Dissembler* and the distribution of data across the channels.

The *Assembler* receives packets from each channel and populates the *Forwarding Buffer*, which contains the reassembled (unscrambled) data to be forwarded to the receiving application.

Figure 1 shows the structure of the network showing the *Dissembler* on the left, followed by an arbitrary number of intermediary nodes, representing other hops along the network. Intermediary nodes are abstracted from an arbitrary number of hops into a single node for purposes of simulation. The intermediary nodes act as a listener for diagnostic purposes, or to simulate injection attacks on the traffic. Finally, the *Assembler* is on the right which takes the multiple inputs and reassembles the data. The source and sink simply represent the applications which produce and receive the data that is being transmitted.

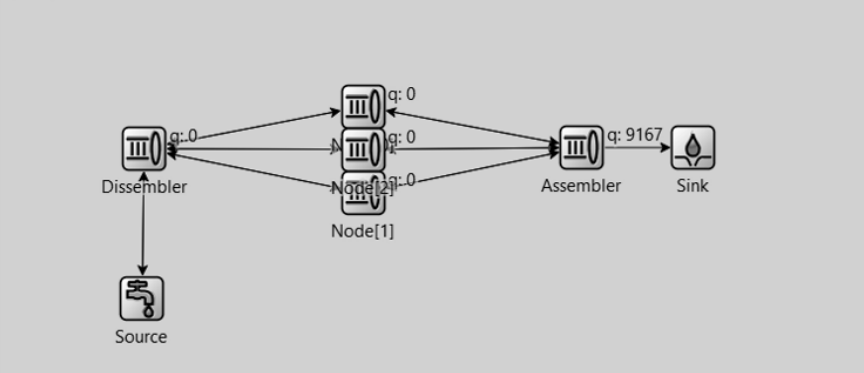

Figure 1: Network Diagram

3.3 The Dissembler

The *Dissembler* creates individual *Channel Packets* to be sent along each channel. As bits stream from the application, the *Dissembler* creates *Data Packets* which contain the unique data to be distributed across the three channels. The size of the *Data Packets* is based on the size of the *Channel Packets,* the number of channels in use and the Duplication Factor (DF) as follows:

$$\textit{Data Packet Size} = \textit{Channel Packet Size} \times (\textit{Number of Channels}/\mathrm{DF})$$

For simplicity, a *Channel Packet* size of 64 data bits was chosen. Any empty space left by data that could not completely fill the last *Data Packet* is padded with 0's in order to maintain the *Channel* and *Data Packet* sizes. While trivially small *Data Packet* and *Channel Packet* sizes were chosen, this approach is easily scalable to larger packets, closer to the Ethernet Maximum Transmission Unit (MTU) of 1500 bytes.

3.4 Data Packets and Channel Packets

For each set of 96 bits that arrive at the *Dissembler*, populating the *Data Packets*, the *Dissembler* distributes the data into specific *Channel Packets* for each channel. *Channel Packet* A1 receives 2/3 of the bits from the *Data Packet* 1. Similarly, *Channel Packets* B1 and C1 receive 2/3 of the bits from *Data Packet* 1. Figure 2 illustrates the specific bits that go into each of the *Channel Packets*, ensuring that each bit is placed on exactly two of the three channels, thus creating a duplication factor of 2.

Data Bit	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Chan A									0	1	0	0	1	0	0	1	0	1	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	0
Chan B									0	1	0	0	1	0	0	1	0	1	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	0
Chan C									0	1	0	0	1	0	0	1	0	1	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	0
Bits A	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0
Data Index									0	2	4	3	8	7	9	11	12	14	16	15	20	19	21	23	24	26	28	27	32	31	33	35	36	38	40	39	44	43	45	47	48	50	52	51
Bits B	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	1	0	1	1
Data Index									1	0	5	4	6	8	10	9	13	12	17	16	18	20	22	21	25	24	29	28	30	32	34	33	37	36	41	40	42	44	46	45	49	48	53	52
Bits C	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	0	1
Data Index									2	1	3	5	7	6	11	10	14	13	15	17	19	18	23	22	26	25	27	29	31	30	35	34	38	37	39	41	43	42	47	46	50	49	51	53
Msg Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43

Figure 2: Channel Packet Data Layout

Figure 2 also illustrates the structure of the *Data Packets* and *Channel Packets*. The “Data Bit” row is the index of the data from the *Data Packet*. The “Chan” rows are populated with the actual bits inside of *Data Packet* 1; note the *Data Packet* is the same across all three channels as all three channels are simply sending different parts of the same message. A particular cell highlighted in a “Chan” row indicates that bit is sent on the respective channel.

Figure 2 also illustrates the data scrambling that occurs within the protocol. This can be seen in the "Data Index" rows that correspond to each "Bits" row. The *Data Packet* bits are placed into the *Channel Packets* "Round Robin" but not sequentially. This means that the n^{th} bit of the *Channel Packet* does not necessarily correspond to the n^{th} bit of the *Data Packet*. Additionally, the first 8 bits are policy header bits and unrelated to data.

The final *Channel Packet* is 64 bits of headers and footers with 64 bits of data. The first 8 policy bits simply state the algorithm used by the *Dissembler* for selecting which bits were placed on which channel, as well as the scrambling code that was used. The packet number simply helps the *Assembler* ensure the correct *Channel Packets* are being assembled together. Finally, the Cyclic Redundancy Check (CRC) is a standard CRC used to check for packet validity.

	8 Bits	64 Bits	8 Bits	8 Bits	8 Bits	10 Bits	22 Bits
1	policy	data A	CS1:B[0:31]	CS2:B[32:63]	CS3:C[0:63]	packet #	CRC
2	policy	data B	CS1:C[0:31]	CS2:C[32:63]	CS3:A[0:63]	packet #	CRC
3	policy	data C	CS1:A[0:31]	CS2:A[32:63]	CS3:B[0:63]	packet #	CRC

Figure 3: Channel Packet Structure [1]

3.5 Checksums

The checksum layouts may appear peculiar at first but serve a specific purpose. The layout is based on previous work that developed this specific checksum and CRC structure so as to detect malicious injection on a channel (Simon 2022). Using this structure together with a DF of 2, data is recovered well. Figure 3 illustrates the layout of the checksums such that each channel carries a checksum for the other two channels. This allows data to be cross-checked multiple ways to ensure validity and, in the event of an error, identify invalid data and use valid data from an uncorrupted channel. This allows the *Assembler* to recover data with minimal need for retransmission.

3.6 The Assembler

The role of the *Assembler* is to take the data from each of the *Channel Packets* and reassemble the data within the *Forwarding Buffer* in order to pass the data to the application. To accomplish this, the *Assembler* must receive packets from each channel and process them. Processing the packets includes confirming packet validity, unscrambling each packet and populating the *Forwarding Buffer* with the appropriate data. If a packet is invalid, the *Assembler* attempts to recover the missing or corrupted data.

3.7 *Assembler* Operation

The *Assembler* will process the *Channel Packets* according to one of two methods. The first assumes no corruption within the channel packets and all data is valid. The second assumes corruption exists within one or more *Channel Packets.*

When the *Assembler* receives data, it assumes data is valid until proven otherwise. This means the *Assembler* waits for at least two corresponding *Channel Packets* to arrive before beginning to process the data. The *Assembler* must only wait for two of the three channels due to our duplication factor of 2, which ensures that 100% of the data is contained within just two of the channels. This assembly without all three channels is referred to as *Speculative Assembly.* When two *Channel Packets* have arrived, the *Assembler* checks the CRCs and Checksums to ensure validity of packets. Assuming data is valid, the assembler places the unscrambled data from both of the channel packets into the *Forwarding Buffer* and continues operation with the next packet in the queue. If *Speculative Assembly* was performed, the unused *Channel Packet* is dropped once it arrives at the *Assembler*.

In the event that the *Assembler* detects invalid data, assembly halts at that point until all three channels have arrived. Without all three channels, the probability of successfully recovering data is low, therefore, *Speculative Assembly* is not attempted. Once data from all three channels have arrived, the checksums are compared in order to determine the corrupt channel and utilize the data from the uncorrupted channels. In the event that no channel can be definitively identified as the corrupt channel, retransmission would be necessary. For purposes of this simulation no feedback is implemented so this need for retransmission was captured as corruption in the final message.

3.8 Metrics for Measuring Performance

The primary testing goals of this protocol are to analyse the levels of corruption in the final message to determine recovery effectiveness of this protocol as compared to a single channel system with no error correction. Additionally, simulation execution time is analysed to assess the performance gain certain features provide the protocol, such as the speculative assembly.

3.9 The Simulation Environment

OMNeT++ is the simulation environment used to implement this protocol. Much of the infrastructure was created by Paul Simon in (Simon 2022). This includes the networking and setup of the simulation environment. All previous sections within Chapter 3 were implemented within OMNeT++ and C++.

4. Analysis

A series of experiments are executed to analyse the performance of the protocol. The results of a Single Channel systems performance, are first, followed by results for a multi-channel setup. In conclusion, we compare the two and discussion the advantages and drawbacks of each.

4.1 Single Channel Performance

Single channel performance is not a primary focus of this protocol; however, it provides a necessary benchmark. Thus, analytical results have been created to compare and benchmark the multi-channel protocol as shown in Table 1. The table shows that without any of the data recovery or redundancy that exists within the multi-channel protocol, the amount of corruption in the final message will be equal to the probability of corruption on the channel.

Table 1: Single Channel Corruption

Channel Corruption	Final Message Corruption
0	0
10	10
20	20
30	30
40	40
50	50
60	60
70	70
80	80
90	90
100	100

4.2 Multi Channel Performance

For multi-channel performance, we begin by looking at the timing performance gains provided through the use of a three-channel system with the speculative assembly feature enabled.

Table 2 shows the timings for a three-channel system with no corruption. The delay simulated for each packet along each channel is listed within the respective ``Delay'' columns. The Speculative Finish and All Finish columns simply report the simulation time at which the message is received. Speculative Finish is when the program completes with speculative assembly, while All Finish is when the program completes after waiting for all packets to arrive. The difference column simply highlights the gain that speculative assembly was able to provide for the protocol.

Table 2: **Three Channel Timings**

Delay			Speculative Finish	All Finish	Difference
A	B	C			
0	0	0.001	20.953	23.388	2.435
0.001	0.001	0.001	-	23.388	-
0	0.001	0.001	-	23.388	-
0.002	0.002	0.002	-	28.002	-
0.001	0.002	0.002	-	28.002	-
0.001	0.001	0.002	23.388	28.002	4.614

Table 2 shows the performance gains achieved with speculative assembly as compared to waiting for data to arrive on all channels. This demonstrates the functionality of the protocol and suggests that as potential delays along channels increase, the amount of time saved by conducting speculative assembly greatly increases as well. This is essential because when conducting a multi-channel communication session, there is no guarantee that every channel will perform as well as the others.

Table 2 illustrates the ability for this protocol to handle complete channel failure and how speculative assembly works when the system is experiencing some level of corruption. For the example shown in Table 2, Channels A and B are each unobstructed, but Channel C is fully corrupted. No data from Channel C arrives at the *Assembler*, yet the protocol successfully reconstructs the data.

Table 3: **Three Channel System with Single Channel Failure**

Corruption			Delay			Speculative Finish	All Finish	Message Corruption
A	B	C	A	B	C			
0	0	100	0	0	0	-	20.953	0
0	0	100	0	0	0.001	20.953	23.388	0
0	0	100	0	0.001	0	-	23.388	0

These examples illustrate 1) how the protocol is effective at increasing availability of the system by remaining resilient during complete channel failures and 2) the performance gains by requiring no packet re-transmissions as a result of this channel failure.

Table 4 presents levels of corruption in the final message with corruption on all channels. With low overall channel corruption, the protocol produces final message corruption that is lower than the overall channel corruption, however when the corruption is high, the overall message corruption is higher than the channel corruption.

Table 4: **Three Channel Corruption**

Corruption			Message Corruption
A	B	C	
10	10	10	3.27
20	20	20	7.5
30	30	30	24.2
40	40	40	35.8
50	50	50	49.4
60	60	60	65.6
70	70	70	78.6
80	80	80	90
90	90	90	97.6

Figure 4 shows the corruption values from Table 1 and Table 4 plotted together. This demonstrates an interesting interaction between the two. When overall corruption levels are low on a network or channels, less than 50%, then the multi-channel system performs much better, having improvements between 1% and as much as 13%. However, when the channel corruption rises above 50%, the multi-channel system experiences a similarly worse performance over the single channel system.

Figure 4: Single-Channel vs Three-Channel Corruption

This relationship makes sense analytically, since the protocol requires at least two channels to transmit successfully. As the probability of corruption increases, the chances of at least two channels being uncorrupted decreases. However, as the probability of corruption decreases, the chance of at least two channels being uncorrupted increases. This results in the inverse relationship demonstrated.

This analysis demonstrates areas and situations where a multi-channel setup is superior to a single channel setup. When corruption levels are greater than 0 but less than 50%, the multi-channel setup is superior to a single channel setup with regards to the success of data transmission. In a realistic scenario where feedback was implemented, this would also result in fewer re-transmissions required. While it was shown that, without corruption, the single channel system is superior to the multi-channel from a bandwidth perspective. Once corruption is involved and the single channel system would need to retransmit 5% to 10% more packets than the multi-channel, the multi-channels' benefits emerge.

Additionally, the additional security benefits of data scrambling and dropout come at no cost when implementing a protocol across three channels with a duplication factor of two. Should an adversary attempt to eavesdrop on the transmissions of a single channel, not only will the bit level scrambling make the data unreadable without the scrambling code, but the missing data will make determining this scrambling code significantly more difficult. In order to reasonably determine the whole message; an adversary would need to sniff on at least two of the channels in order to have all of the data to decode. However, the added problem of too much information also complicates the problem. Sorting out the “chaff” from the unique data would require significant time and effort, and this comes at no additional cost to the user, aside from the standard multi-channel overhead.

Finally, the overall availability of the multi-channel system is clearly superior to a single channel system. With a single channel, if the channel fails, then the entire system will not work. A channel may fail for many reasons, from something benign like maintenance on a channel or a cable being accidentally cut, to something malicious like a Denial of Service or Distributed Denial of Service attack. Regardless of the cause of an outage, a single channel system would be unable to operate until the cause of the outage has been resolved, or some alternative has been worked out. With the multi-channel protocol, a failure of a single channel for any reason is completely abstracted away from the user and application. They would be completely unaware of the issues faced by the network because there would be, at most, a slight increase in the delay the network traffic experiences; as demonstrated through Table 2.

4.3 Results Summary

Overall, it has been shown that a multi-channel protocol as described in this research is able to produce lower overall corruption in a contested environment, assuming corruption levels are below 50%. Additionally, the benefits of speculative assembly have been shown within the context of the protocol described. While some drawbacks to the protocol exist, when operating within a contested environment, the security and performance benefits of the protocol illustrate the benefits of using a multi-channel system.

5. Conclusion

While there are numerous benefits to adopting a multi-channel strategy such as the one described within this research, there are also numerous reasons to avoid such a strategy. This section examines what was shown within this research, the significance of the findings, and suggests opportunities for future work.

5.1 Research Conclusions and Significance

The primary contribution of this research is demonstration of the benefits of a multi-channel protocol when operating in a contested environment at less than 50% channel corruption. Under these conditions, the data corruption experienced by the final message is anywhere from 5% to 13% less than a comparable single channel system.

Additionally, the security benefits provided by a multi-channel protocol that implements data dropout and scrambling is clear. The challenges to an eavesdropper attempting to read the original message are significant due to the removal of information available to the attacker. Together with scrambling, this data obfuscation results in significantly increased security over a single channel system with lower processing and overhead than encryption.

The development of this simulation environment and protocol creates the opportunity for further research into multi-channel protocols and further exercising this particular protocol to explore additional benefits or features.

5.2 Future Work

The primary area for future work is advancing features of the protocol within the simulation environment, as well as to more thoroughly exercise the simulation in order to better characterize the performance and abilities of the protocol.

The implementation of an initialization vector or handshake would be valuable. The way to define and coordinate channels would be essential for smooth operation of the protocol. Additionally, in order to maintain the security of the protocol, this initialization must happen in a discreet way so that adversaries are unable to gain information regarding where additional channels are, what protocols are being used or any other information that could be beneficial towards the deciphering of the messages. One potential solution would be to utilize PKI to encrypt the initial handshake to ensure it remains secret, then use the protocol as described for the remaining data transfer, so as to not incur the overhead of encryption with the remaining data transfer.

Exercising the simulation more thoroughly would also be a beneficial exercise to generate more data on the performance of the protocol in specific edge cases or with larger data sets. Combining this with a feedback and re-transmission capability would facilitate a realistic comparison to a single channel system.

5.3 Conclusion

While no perfect solution can ever exist in a complex and ever-changing problem, this research seeks to create a viable solution for a specific scenario by utilizing multi-channel communications. Through the use of data obfuscation via bit scrambling and data dropout, the challenges of a potential attacker have been made significantly greater. Additionally, performance gains in non-permissive environments have been demonstrated, showing how a multi-channel protocol is superior to a single channel when data corruption rates are below 50%. Finally, practical applications and areas for future focus and work have been discussed to illustrate the significance of this research and the way for it to move forward. Future work could include the progression of the simulation to better encompass a robust and complete protocol, or implementation within a network that has the opportunity to utilize multiple channels.

References

Bao, S.-D., Chen, M., & Yang, G.-Z. (2017). A Method of Signal Scrambling to Secure Data Storage for Healthcare Applications. IEEE Journal of Biomedical and Health Informatics, 21(6), 1487–1494. https://doi.org/10.1109/JBHI.2017.2679979

Bonaventure, O., & Seo, S. (2016). Multipath TCP Deployments. IETF.

Dmitrienko, A., Liebchen, C., Rossow, C., & Sadeghi, A.-R. (2014). On the (In)Security of Mobile Two-Factor Authentication. In N. Christin & R. Safavi-Naini (Eds.), Financial Cryptography and Data Security (pp. 365–383). Springer Berlin Heidelberg.

Ford, A., Raiciu, C., Handley, M., Bonaventure, O., & Paasch, C. (2020). TCP Extensions for Multipath Operation with Multiple Addresses (RFC No. 8684; pp. 1–68). RFC Editor. https://www.rfc-editor.org/rfc/rfc8684.txt

Hughes, G. F., & Murray, J. F. (2005). Reliability and Security of RAID Storage Systems and D2D Archives Using SATA Disk Drives. ACM Trans. Storage, 1(1), 95–107. https://doi.org/10.1145/1044956.1044961

Modarressi, A. R., & Skoog, R. A. (1990). Signaling System No.7: a tutorial. IEEE Communications Magazine, 28(7), 19–20. https://doi.org/10.1109/35.56239

Simon, P. (2022). Development of a Security-Focused Multi-Channel Communication Protocol and Associated Quality of Secure Service (QoSS) Metrics [Phdthesis]. Air Force Institute of Technology.

Simon, P. M., Graham, S., Talbot, C., & Hayden, M. (2021). Model for Quantifying the Quality of Secure Service. Journal of Cybersecurity and Privacy, 1(2), 289–301.

What is OMNeT++? (2022). https://omnetpp.org/intro/