

Santa Clara University

Scholar Commons

Computer Science and Engineering Senior
Theses

Engineering Senior Theses

6-10-2020

ACAS: Always Connected, Always Secure

Blaise Aranador

Antonio Gigliotti

Shining Liu

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior



Part of the [Computer Engineering Commons](#)

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 10, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

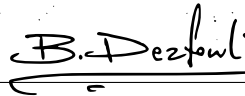
Blaise Aranador
Antonio Gigliotti
Shining Liu

ENTITLED

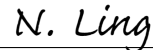
ACAS: Always Connected, Always Secure

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

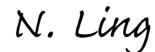
BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING
BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING



Thesis Advisor



Department Chair



Department Chair

ACAS: Always Connected, Always Secure

by

Blaise Aranador
Antonio Gigliotti
Shining Liu

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science in Computer Science and Engineering
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 10, 2020

ACAS: Always Connected, Always Secure

Blaise Aranador
Antonio Gigliotti
Shining Liu

Department of Computer Science and Engineering
Santa Clara University
June 10, 2020

ABSTRACT

There is no place where safety is more important than in the home. Research has shown that home security systems are effective in deterring burglars; additionally, these security systems allow residents to monitor their property at all times, even while they are away. More and more of these home security devices rely on a stable Internet connection and cannot provide functionality without it. ACAS is a system that helps keep smart devices connected to the Internet, even in the event of a home internet outage.

ACAS includes a programmable router that can connect to multiple Internet sources, which sets it apart from other routers on the market. ACAS can connect to two or more Internet sources at a time and then broadcast a wireless Internet signal that one's smart security devices (and any other device) can connect to. The router uses one Internet source at a time to provide a wireless signal for all devices to connect to it, but in the case that the Internet source goes down for any reason, ACAS automatically switches to one of the other Internet sources connected to it. This provides a reliable backup and keeps devices connected to the Internet as long as one of the multiple Internet sources connected to the router is up and running.

Our system also includes a web application that provides the ability to configure some aspects of the router and obtain up-to-date statistics about the workings of the router itself. Users can check the network speed of the Internet connection and choose which of the multiple Internet sources is the main Internet source at any given time.

Acknowledgments

We would like to thank Dr. Behnam Dezfouli for his guidance and instruction throughout this entire process. His advice and recommendations every step of the way helped us tremendously.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Solution	1
1.3	Existing Solutions	2
1.3.1	Hardware Load Balancer	2
1.3.2	Cellular Data	2
2	Requirements	3
2.1	Functional	3
2.1.1	Provide Internet in Case of Outage	3
2.1.2	Open Source	3
2.1.3	User Configuration and Monitoring	3
2.2	Non-functional	3
2.2.1	Ease of Use	3
2.2.2	Affordability	3
2.3	Design Constraints	4
2.3.1	Internet Bandwidth	4
2.3.2	Network Interface Card Requirements	4
2.3.3	Wireless Network Interface Source	4
3	System Architecture	5
3.1	Description of System Implemented	5
4	Technologies Used	7
4.1	Hardware	7
4.1.1	Router: Kingdel Powerful Mini Desktop Computer	8
4.1.2	Access Point: Raspberry Pi 4 Model B	8
4.2	Software	8
4.2.1	Open vSwitch	8
4.2.2	OpenFlow Protocol	8
4.2.3	NGINX Web Server	8
4.2.4	Python	8
4.2.5	Django	8

5	Design Rationale	9
5.1	Hardware Design	9
5.1.1	Kingdel Powerful Mini Desktop Computer	9
5.1.2	Raspberry Pi 4 Model B	9
5.2	Back-end Design	10
5.2.1	Open vSwitch	10
5.2.2	OpenFlow Protocol	10
5.2.3	Web Application	11
5.3	Front-end Design	12
5.3.1	Web Application	12
6	Testing	14
6.1	Plan	14
6.2	Results	14
7	Societal Issues and Considerations	16
7.1	Ethical	16
7.2	Social	16
7.3	Political	16
7.4	Economic	17
7.5	Health and Safety	17
7.6	Manufacturability	18
7.7	Sustainability	18
7.8	Environmental Impact	18
7.9	Usability	18
7.10	Lifelong Learning	18
7.11	Compassion	19
8	Conclusions	20
8.1	Obstacles Encountered	20
8.2	Suggested Changes and Development	20
8.3	Lessons Learned	21
	Appendix A: User Manual	22
	Appendix B: Installation Guide	24
	Appendix C: Hardware Parts and Specifications	30
	Appendix D: Source Code	31
	Appendix E: Senior Design Presentation	34
	References	40

List of Figures

3.1	Architecture Diagram	5
4.1	System Components	7
6.1	Internet Downtime When Disconnecting One Internet Source	15

List of Tables

7.1 Itemized Price of Components 17

Chapter 1

Introduction

1.1 Motivation

Safety is paramount, and there is no place where safety is more important than in the home. According to a 2017 crime statistics study conducted by the Federal Bureau of Investigation (FBI), there is a home burglary every 22.6 seconds in the US; on top of that, the average burglary victim loses over \$2000 in goods [9]. Research has shown that home security systems are effective in deterring burglars, as “83% [of burglars] admitted that they specifically look to see if there’s an alarm” and “60% would change their mind if there was one installed” [2]. These security systems also allow residents to monitor their property at all times and to contact the police if necessary, even while they are away. Currently, many security systems are integrating smart devices that rely on an Internet connection to inform residents in near real-time. However, the reliability of the Internet connection depends on the provider, electrical company, and the wear and tear of cables. Without an Internet connection, many of these devices lose core functionality.

1.2 Solution

Since all of these smart devices require an Internet connection to be useful, we created a smart connection monitoring and recovery agent that can be added to the home network to monitor connectivity and, in the case of disconnection, connect the smart devices to a backup Internet source.

The smart connection monitoring agent is designed and implemented using a Kingdel Powerful Mini Desktop Computer, which acts as a middle-man to facilitate reliable connections between smart devices and the network it connects to. It automates a majority of the connection process so the user does not have to be concerned with technical details. The PC is configured as a router and provides wireless internet to all devices connected to it. The PC also relies on multiple Internet sources to broadcast this connection.

The system provides an accompanying web application that allows users to manage preferences if they are more

familiar with the technical details of the connection process. The application can also be used to give relevant statistical data, such as network speed and status of Internet connections.

Furthermore, the solution we created provides a much more ideal fallback plan when Internet connection is lost compared to current existing solutions. Overall, our product ensures that smart home security systems are more reliable and more secure, thus offering greater safety and protection of persons and property.

1.3 Existing Solutions

1.3.1 Hardware Load Balancer

A load balancer is a hardware device that can distribute network traffic to a number of connected devices [10]. These load balancers are traditionally used in cloud computing and to distribute this network traffic to servers in a data center. This device can be used in a smaller scale as well and would be able to take multiple internet sources and distribute this traffic to the connected devices.

The downside of load balancers are that they are expensive and do not provide the ability to choose the amount of Internet sources connected to the device itself [13]. Load balancers have a specific amount of Ethernet ports connected to them and there is no way to add more ports if more connections are needed. They are also costly as they are traditionally used in large scale cloud computing networks and therefore built with high end hardware that is not cheap and not accessible to the average individual.

1.3.2 Cellular Data

Cellular data is another backup plan option for an Internet outage. There are some services that provide backup cellular data to your home in the event of a home Internet outage.

While cellular data does do a good job at providing a reliable backup plan, it has its drawbacks as well. For starters, this requires additional hardware to be installed and configured in one's home [3]. Cellular data is also not free, as it usually requires users to pay a monthly or yearly subscription for the services, and this can add up quickly over time [3]. Lastly, cellular data traditionally provides a much slower internet connection than what is provided by ISPs for home networks.

Chapter 2

Requirements

2.1 Functional

2.1.1 Provide Internet in Case of Outage

In the case of an Internet outage, all devices that are connected to the router must be able to regain Internet connection. This connection comes from a backup Internet source (different than the home network).

2.1.2 Open Source

The proposed solution must be open source so that anyone can easily replicate or improve upon the work done and build this solution for themselves.

2.1.3 User Configuration and Monitoring

Users of the system must have the ability to choose the Internet sources that are used for the router and have control over which network to use at any given time. The user also needs to be able to see network statistics, such as bandwidth, and the current configuration of the system.

2.2 Non-functional

2.2.1 Ease of Use

The system should be easy for users to navigate and configure. The web application should be straightforward and should require little, if any, technical knowledge to understand or navigate.

2.2.2 Affordability

The system should be affordable to consumers, particularly in comparison to competing load balancers and cellular data plans, and especially as a potential one-time purchase.

2.3 Design Constraints

2.3.1 Internet Bandwidth

Our solution uses a Raspberry Pi 4 as the access point for the system. Although the Raspberry Pi provides great functionality for the price, the functionality is limited at times. The network card that is in the Raspberry Pi can only provide a certain speed when providing Internet access to the devices connected to it [17]. This can lead to slower network speeds because of the hardware being used.

2.3.2 Network Interface Card Requirements

The proposed solution can be run on all hardware, but that hardware must have a Network Interface Card (NIC) that is capable of broadcasting a wireless signal.

2.3.3 Wireless Network Interface Source

Open Virtual Switch is not compatible with wireless network interfaces being connected to its bridges and because of this, a workaround is needed. Our solution requires a separate piece of hardware to connect to a wireless Internet source and then feed that signal to the OvS bridge. In our implementation, that piece of hardware is a Raspberry Pi 4 [23].

Chapter 3

System Architecture

3.1 Description of System Implemented

A high-level overview of our solution is shown in Figure 3.1. The ACAS system is composed of several parts. The main component is our Kingdel Powerful Mini Desktop Computer, which acts as our router. There are several network interface ports on the ACAS system that are connected to different internet sources.

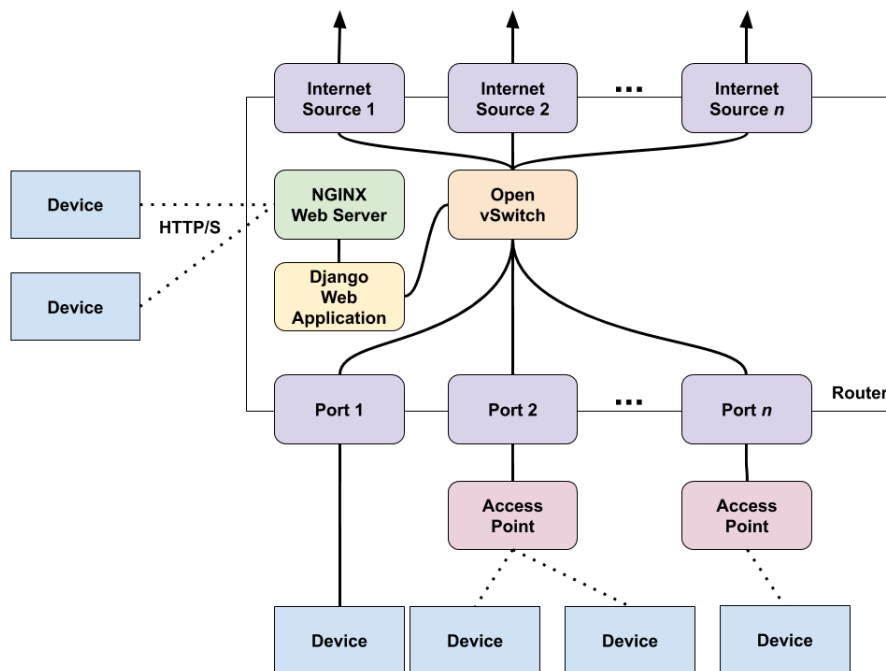


Figure 3.1: Architecture Diagram

The router has Open vSwitch installed, which allows us to build a network bridge and add ports and bonds to the

bridge so that it is possible to switch between the different Internet sources that a user's device might be using. The bond connects the multiple Internet sources together to act as a single port for a user's devices to use. Ports are added to the Open vSwitch bridge to connect a user's devices to the internet source being used.

Users can connect to the ACAS system through either Ethernet or Wi-Fi. If Wi-Fi is the preferred choice of connection, an access point built with the Raspberry Pi 4 Model B board must be used to provide Wi-Fi capabilities [15].

The ACAS system currently has the NGINX web server and web application running on the system itself. A user's device can connect to the web application by communicating with the NGINX web server via HTTP/S traffic [30].

The web application uses Django, which is a Python based web application framework [6]. The ACAS web application utilizes Python's available modules [20] to provide network and system statistics as well as the ability to configure the system. The speedtest-cli module is used to obtain upload and download speeds of the Internet source the ACAS system is currently using [16]. The web application also utilizes Python's ability to run shell commands on the system to obtain information and configure the Open vSwitch bridge's bond.

Chapter 4

Technologies Used

4.1 Hardware

The system created requires a few forms of hardware to function correctly. The access point, router, and Wi-Fi sources in the system are three separate pieces of hardware that all connect to one another. Figure 4.1 demonstrates how these components are wired together to reflect the system architecture in Figure 3.1. The following sections describe each piece of hardware in more detail.

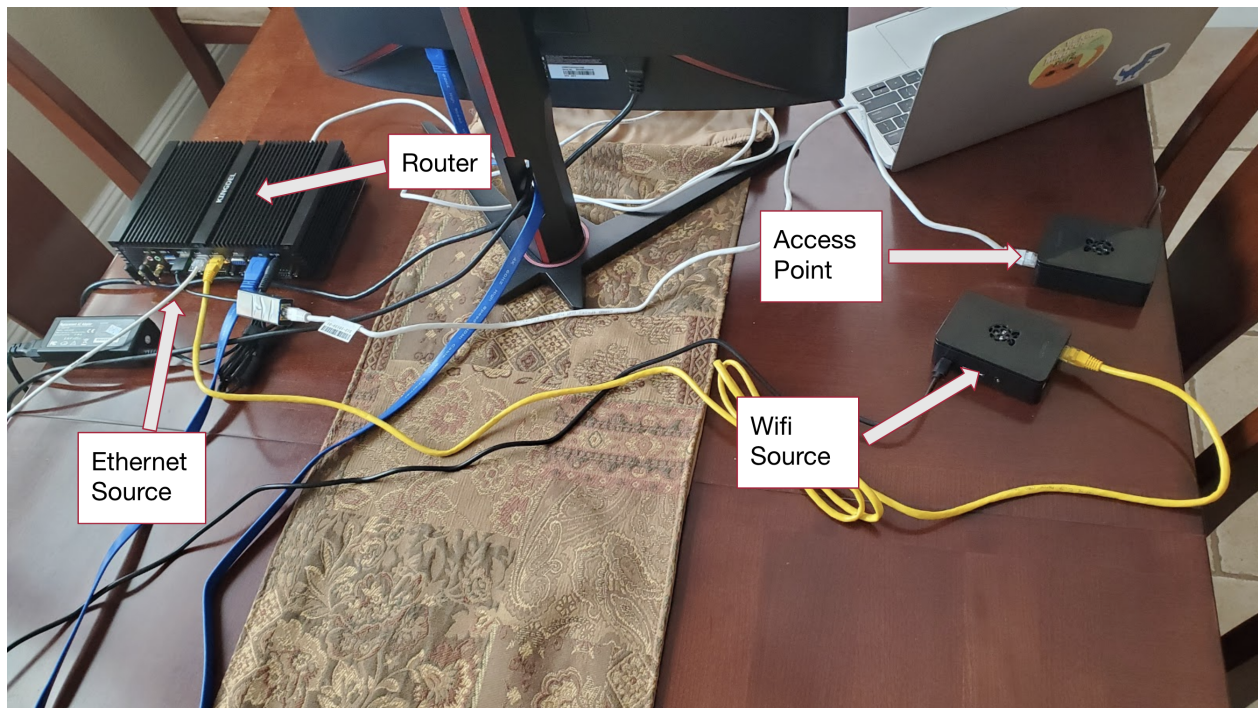


Figure 4.1: System Components

4.1.1 Router: Kingdel Powerful Mini Desktop Computer

The Kingdel Powerful Mini Desktop Computer is used to build a router that runs the Open vSwitch software, Open-Flow Protocol, an NGINX web server, and a Django web application.

4.1.2 Access Point: Raspberry Pi 4 Model B

A Raspberry Pi 4 Model B is used as the access point for the router and therefore provides Internet connection to any device connected to it [15]. A second Raspberry Pi 4 Model B is used as a wireless Internet source for the router [23].

4.2 Software

The following sections describe the software components used in the system.

4.2.1 Open vSwitch

The Open vSwitch software is used to connect the user's home devices to the different Internet sources. The two main aspects of OvS that are used are bridges and bonds. OvS bridges connect multiple physical network interfaces on the PC. OvS bonds allow two or more network interfaces to be treated as one port [25].

4.2.2 OpenFlow Protocol

The OpenFlow protocol controls network flow throughout the PC and more specifically across the OvS bridge [27]. This protocol ensures that the network traffic flows from the Internet sources to the access point in our system.

4.2.3 NGINX Web Server

The NGINX web server is used to connect the user to our web application via HTTP/S [30].

4.2.4 Python

Python is a powerful programming language [21] that utilizes modules [20] that are used in the ACAS system to run shell commands to gather system and network statistics as well as configure the router's Open vSwitch settings [16].

4.2.5 Django

Django is a Python-based web framework [6] that is used to build our web application to gather system and network statistics, as well as configure the ACAS system.

Chapter 5

Design Rationale

5.1 Hardware Design

The hardware used for the system was chosen such that a reliable system could be created at a reasonable cost for a consumer.

5.1.1 Kingdel Powerful Mini Desktop Computer

The Kingdel mini PC is used to run OvS and OpenFlow and control packet flow throughout the system. While many hardware systems can handle this work load, the Kingdel PC is powerful enough to handle all of this without any issue and allows for a more scalable design over something with much less power. The Kingdel PC also has an x86 Intel CPU architecture that allows the PC to run Linux, more specifically Ubuntu, which was necessary for the system. OvS and OpenFlow both run on Linux based machines and the Kingdel PC was able to provide the correct operating system.

5.1.2 Raspberry Pi 4 Model B

A Raspberry Pi 4 Model B is used in two different ways in the system. One use case is as the access point [15]. The Raspberry Pi provides a wireless network signal to all devices connected to it. A Raspberry Pi 4 was chosen for this application because of the price to performance ratio. While a Raspberry Pi 4 is not the most powerful machine, it is able to provide a reliable Internet connection for a reasonable price. The second application of a Raspberry Pi 4 Model B is for a wireless network source for the router [23]. This Raspberry Pi connects to the Internet and in turn, provides a connection to the second Raspberry Pi that then outputs that network signal. In this case, a Raspberry Pi was also chosen because of its price to performance ratio, as it is able to provide the required performance for a reasonable cost [17]. Furthermore, Raspberry Pis have a Linux based operating system that allowed for the boards to be used for both applications within the system. See Appendix B on how to setup the Raspberry Pi 4s for the system.

5.2 Back-end Design

The back-end components of the ACAS system were chosen so that the system statistical information and configurations can be delivered to the user and system efficiently and effectively.

5.2.1 Open vSwitch

Open vSwitch is an open source virtual switch software that runs on Linux and provides control of packet switching within the operating system [5]. OvS was chosen to provide the packet switching over Linux switching features for a few main reasons. First of all, OvS is used in large scale productions and is therefore scalable and proven to handle large workflows. OvS also provides the ability to easily see network statistics and status on the switch itself. OvS also provides a bond feature that was instrumental in the completion of the system, but this was not the only OvS feature used. The two main features of OvS that were used are bridges and bonds [28]. See Appendix A on how to install Open vSwitch on any x86 machine.

Bridge

An OvS bridge is a virtual network bridge that creates a single aggregate connection between interfaces connected to bridge. This allows for network to flow from one network interface to another without any issue. This is what allowed the system to forward packets from one interface to the next. The bridge allows as many network interfaces as the hardware provides to be added to the bridge which allows the system to scale [24]. See Appendix A on how to work with OvS bridges.

Bond

An OvS bond allows two or more network interfaces to share network traffic and act as a single port [25]. This allows our router to have multiple Internet sources connected to a bond at any given time. OvS bonding also has protocols attached to each bond that control the balancing of the bond [19]. The bond mode that was used in our system was the active backup protocol. This allows the bond to use one of many Internet sources at a time and switch over to another Internet source in the event of an outage. See Appendix A on how to work with OvS bonds.

5.2.2 OpenFlow Protocol

OpenFlow is a programmable network protocol that is designed to control network traffic across a router and/or switch. OpenFlow runs on top of OvS and consists of three parts: flow tables, a controller, and an OpenFlow protocol [7]. The flow tables are setup on the switch itself and define what happens to packets when they enter/exit a switch. A controller is the software that talks to the switch, and the OpenFlow protocol is what enforces certain rules on network flow within a flow table [7].

All of these parts were used in ACAS in conjunction with OvS. While OvS connects network interfaces to ports on the network bridge, OpenFlow controls the network traffic flowing between these ports on the network bridge itself. OpenFlow ensures that the system forwards packets from the OvS bond (internet sources) to the OvS port (access point) and then back the same way for returning packets. Without using OpenFlow and a controller, there would be no way to ensure that packets are flowing correctly across the bridge and would result in an unstable system.

5.2.3 Web Application

NGINX Web Server

For users to communicate with our Django web application, our ACAS system needs to provide a web server. The web server allows users to connect to our web application through HTTP/S traffic [30]. The exchange between HTTP requests and responses allows the user to get information about the ACAS system as well as configure the system's settings. See Appendix B on how to install NGINX Web Server.

The web server we decided to utilize to allow users to connect to our web application is an NGINX web server. We decided to utilize an NGINX web server over an Apache web server because it had similar or better performance statistics when delivering static and dynamic content to a user's web browser. In terms of dynamic content, both NGINX and Apache web servers had similar performance statistics when delivering dynamic content. In terms of static content, an NGINX web server is able to deliver static content about 2 to 2.5 times faster than an Apache web server [14].

Our web application is mainly static content (e.g. background, headers, buttons, etc.) with some dynamic content (i.e. the system statistic information and configuration status messages). With the obvious performance advantage that NGINX has in terms of static content delivery and similar performance statistics delivering dynamic content, we decided to utilize an NGINX web server to deliver our web application to the user.

Django: Python

The ACAS web application was built using Django's Python-based web application framework [6]. Python is a powerful language [21] that allows us to run shell commands on the ACAS system to gather system statistics on the Open vSwitch configuration, specifically the bond used to connect multiple internet sources and the port statuses for each of the internet sources connected to our system. Possible port statuses that are shown on the system are available and unavailable, which means that the internet source is ready or not ready to be used. Another port status that is shown is which port is the active slave. The active slave status is given to the port that the user's home devices are currently

using to get Internet access [12]. See Appendix D on the specifics of how Python was utilized.

Python also allows us to use developed modules that provide network statistics of our ACAS system. One module that we used is the speed test command line interface (speedtest-cli). The speed test module allows us to run speed tests to gather upload and download speeds [16].

With the various information Python is able to gather about the ACAS system, we can then use this information to display it to the user on the web application.

5.3 Front-end Design

5.3.1 Web Application

Django: HTML and CSS

Since Django is used for the back-end of the web application, the front-end follows suit and utilizes Django's built-in template language to create web pages in HTML and CSS. For the purposes of displaying network statistics and port statuses, as well as basic network configuring capabilities, we determined that it was not necessary to use more powerful frameworks like React.

User Experience and Visual Design

Although user experience and visual design were not considered priorities, several efforts were made to ensure the usability of the web application. Open Sans is the font used in the web application, as it is a highly legible and versatile font, particularly suited for prolonged reading on both mobile and web interfaces [11]. It is also suited for use in body text, titles, and other design elements [22]. Additionally, the font is visually simple, professional, and friendly—all characteristics we want associated with ACAS.

The web application also features a monochromatic color scheme with blue as the dominant color. 51% of the most valuable brands and logos use a monochromatic color scheme, and over 35% use the color blue [18]. Blue tends to facilitate a sense of trust, security, and dependability—again, all characteristics that would be beneficial for ACAS to be associated with [18]. Blue also tends to be a universally favored color, commonly liked across genders and age groups [18].

According to WCAG (Web Content Accessibility Guidelines) 2.0, there should be sufficient contrast between text color and background color. The minimum color contrast ratio is 4.5:1, with a preferred color contrast ratio of 7:1 [29]. From an accessibility standpoint, following WCAG guidelines helps people with low contrast sensitivity, which

is common in older people, as well as people with color blindness [1]. There are several additional usability benefits as well; content tends to remain readable under different lighting conditions and becomes more legible for everyone, even for those without specific visual requirements [1]. ACAS's web application adheres by these guidelines, achieving more than the preferred 7:1 color contrast ratio whenever text color and background color are concerned.

These user experience and visual design considerations make ACAS highly suited for extension and elaboration in the future.

Chapter 6

Testing

6.1 Plan

The most important aspect of the system is functioning as the backup plan if a user's home Internet loses connection for any reason. Our system recognizes when a current connection goes down and then switches to a backup connection. This act of switching between networks does not happen instantly—there is a delay between when a network loses connection and when the backup network is used to give Internet to the access point. This Internet downtime delay was tested.

Tests were conducted on the full system with one device connected to the access point of the router. There were two Internet sources connected to the router, one being a home network and the other a public Wi-Fi network. We had a laptop connected to the access point and opened a console and pinged the Internet. We then went to the router and simply unplugged the current active Internet source to simulate an Internet outage. During this time, the laptop continued to ping the Internet and stopped being able to ping when the Internet source was unplugged. After some time, the laptop would be able to ping the Internet once again. The delay when the laptop was unable to ping was timed and the results are shown in Figure 6.1.

6.2 Results

Figure 6.1 shows the results of our experiments. There were ten trials so that an average length of downtime when switching from one Internet source to the next could be calculated. As can be seen from the graph, our system takes somewhere between 15 and 20 seconds to recognize an outage and switch Internet sources, with an average of 17.16 seconds. This slight delay in response time is due to a handful of factors. Firstly, the OvS bond has to recognize that the active connection is down and most likely will not reconnect. The OvS bond also has to determine which of the backup sources (if there is more than one backup) to choose and make sure that this source does indeed have an active

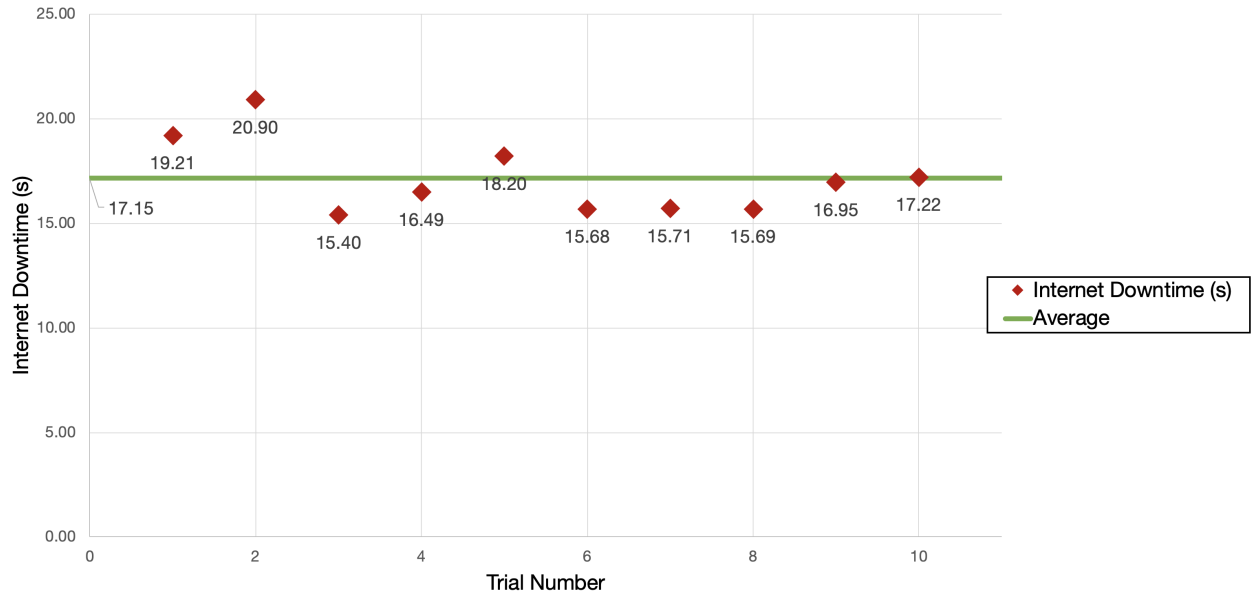


Figure 6.1: Internet Downtime When Disconnecting One Internet Source

Internet connection. Finally, OvS has to do the switching itself and forward all network traffic through the new active source, rather than the original source. All of these factors play a role in the time delay that is seen in Figure 6.1.

Chapter 7

Societal Issues and Considerations

7.1 Ethical

Ethical questions about data and privacy could be brought up in relation to our project. Our team realizes that this is a sensitive subject, but we do believe in protecting consumer rights. As ACAS can be further developed in the future, implementing more robust security strategies should be considered a priority.

7.2 Social

ACAS is intended to eventually become a consumer product. This means that some people may have limited or no access to the benefits of ACAS because they may be unable to afford it. To combat this potential blocker, we have built ACAS to be as affordable as possible so that we can provide this service to as many people as possible. Even now, there are ways to make ACAS more affordable than our current implementation, especially by using cheaper hardware or making use of what consumers probably have lying around the home (e.g. Ethernet cables). For a more detailed discussion on the economics of ACAS, please refer to Section 7.4.

7.3 Political

We see some potential linkage to the debate surrounding net neutrality, particularly since ACAS has the potential to provide aggregate network speeds given more time to develop. This may cause some to call into question if ACAS provides faster network speeds only to people who can afford it. However, we have not built ACAS with the intention of influencing politics and political processes; we simply believe ACAS can have a positive influence on consumer safety, health, and convenience.

Item	Price (USD)	Quantity
Kingdel Powerful Mini Desktop Computer	\$406	1
Raspberry Pi 4 (CanaKit)	\$100	2
Miscellaneous cables and dongles	\$10	2

Table 7.1: Itemized Price of Components

7.4 Economic

ACAS was built with the intention of keeping costs low, or using materials that a consumer is likely to already have. The current implementation of our system has costs per component as shown in Table 7.1.

In total, our implementation cost about \$626 USD. There is notable potential for this price to be significantly lower, such as purchasing the Raspberry Pi 4s by themselves instead of purchasing the kits. We estimate that it is possible to implement ACAS for under \$300.

Something of note, however, is that we have not created any new physical components that require separate manufacturing, nor does any of the software require separate payment. This price point associated with ACAS merely reflects the usage of physical components that are already in production and that a user may already have for other purposes.

7.5 Health and Safety

We see a lot of potential use for ACAS in terms of medical devices. The Internet of Things healthcare market is projected to surpass \$33.7 billion by 2025, and currently, almost a third of smart devices are used in the healthcare industry [4]. With ACAS, we can better ensure that critical health information is passed onto caretakers and healthcare professionals in a timely manner so that patients can get the attention they need faster and with more accuracy.

One concern relating to health and safety is the risk associated with our system going down. According to our testing results in Figure 6.1, there is also some delay in switching from one source to another. How and who should be held responsible if, during those 17 seconds, someone breaks in to steal property or risk the safety of the people living in the residence? This suggests that decreasing downtime could be an area of optimization for future work that could significantly improve the applicability of ACAS in real-world settings.

7.6 Manufacturability

In terms of manufacturability, ACAS is highly flexible for change. The software runs on commodity hardware and can work on any x86 machine. Throughout this project, we have not created any additional physical components. Only hardware that is already in production is used in our implementation.

7.7 Sustainability

ACAS is built with ease of modification in mind and is thus highly sustainable. It utilizes open source software, which means that ACAS can be improved and maintained by others, and that it is free to do so. Because ACAS mainly utilizes existing resources, we also believe that ACAS can successfully exist as part of a sustainable economy.

7.8 Environmental Impact

Through this project, our team has mainly created new software. Thus, it would seem that there is little to no environmental impact on this front. However, the software does require hardware components to run on. Although many types of commodity hardware components can be utilized, these components are typically not recyclable or compostable.

7.9 Usability

In terms of usability, we have designed and built a user interface to accompany the system, so that the user is able to control and configure parts of the system without needing to learn new skills to operate ACAS. Several usability considerations were made in the creation of the web application and are detailed in Section 5.3.1, though there are definitely improvements and functionalities that could be made or added to enhance usability of the web application.

7.10 Lifelong Learning

This project has encouraged the team to study more about computer networking. Previously, this was a topic that the team members found somewhat intimidating, but through ACAS, we have become more comfortable and can envision ourselves in careers that are directly relevant to computer networking in the future.

This project has also helped us practice teamwork and project management skills, which, like most skills, can become better with practice. Due to many unforeseen circumstances and design changes, our team has particularly needed to learn when it is appropriate to ask for help, and when it is appropriate to pivot. While inexperience in these aspects caused overall delays to our progress, we feel that these are valuable lessons learned that we will take with us as we enter the workforce.

7.11 Compassion

As aspiring engineers educated at Santa Clara University, we aim to be sensitive to the needs of people. We see many potential issues that could be mediated with ACAS, especially in terms of healthcare. ACAS, especially when further developed, can better ensure that caretakers and healthcare professionals are notified swiftly and accurately about the health status of patients, potential patients, or high-risk patients, especially related to adverse events such as heart attacks, strokes, seizures, and falls, thus minimizing suffering and potential loss of life.

Chapter 8

Conclusions

8.1 Obstacles Encountered

In working on this project, we have encountered several obstacles. These can be largely categorized into three categories: unforeseen circumstances, design changes, and time management.

- **Unforeseen Circumstances:** Some unforeseen challenges we encountered included a surprising lack of documentation and resources for OvS, as well as the sudden shift to remote work that meant we were unable to share hardware resources or simply work together in person. We also consistently experienced issues with shipping equipment and time delays related to that.
- **Design Changes:** We originally wanted to use the Data Plane Development Kit, or DPDK, to bypass the kernel to improve the speed and performance of our system, but DPDK requires high-end hardware that we did not have access to, so we ended up needing to find a workaround. We also needed to use a Raspberry Pi 4 as a Wi-Fi source because adding the on board wireless card to the bridge was not compatible. We also originally planned to create a mobile application and did quite a bit of work on it before deciding that we needed to transition to a web application.
- **Time Management:** We ran into some trouble with managing the time constraints of this project. All team members had priorities and circumstances that often needed to be dealt with more urgently. Due to some of the design changes we decided on, we also needed to order additional equipment, which caused even more delays.

8.2 Suggested Changes and Development

The list below contains some changes and development that could be made to the ACAS system to improve functionality and applicability.

- **Improve performance and decrease network downtime:** To do this, more or different hardware may be required.

- Improve web application: This includes improving the usability of the application, as well as moving it to a virtual machine so that the web application is hosted publicly.
- Develop a mobile application: An equivalent mobile application may prove useful to users, particularly in terms of notifications that could alert the user immediately to changes on the network.
- Scale up the project.
- Utilize network automation tools such as Ansible, Chef, and Puppet to automate updates and configurations of multiple ACAS systems.

8.3 Lessons Learned

Throughout this whole process, our team found things that worked well, things that did not work so well, and things that we would do differently.

- Consistent meetings and check-ins facilitate progress, both in terms of quality and speed.
- Clarifying requirements as much and as early as possible can prevent pain points and the need to pivot later on.
- Starting early means there is more time to research first and thoroughly without feeling rushed to start implementation.
- Be prepared for things to not work out. Have systems and protocols in place so that when the unexpected does happen, pivoting is a much smoother process.

Appendix A: User Manual

Configure Router

The following instructions are to be run in terminal.

1. Create Bridge

- `sudo ovs-vsctl add-br <bridge>`
 - Note: <bridge> is that name for the bridge being created and can be anything

2. Add network interface that connects the access point to the bridge

- `sudo ovs-vsctl add-port <bridge> <network interface 0>`
 - Note: Network interface names can be found using the "ifconfig" command

3. Add bond to bridge

- `sudo ovs-vsctl add-bond <bridge> <bond> <network interface 1> <network interface 2> bond_mode=active-backup`
 - Note: <bond> is the name of the bond being created and can be anything
 - Note: The number of Internet sources that will be used for the router should be added at this point. Continue to add as many network interfaces as needed in the same format as the first two shown here.

4. Configure packet flow

- `sudo ovs-vsctl --columns=name,ofport list interface`
 - Take note of port numbers for all ports
- `sudo ovs-ofctl add-flow <bridge> in_port=<num 1>,actions=<num 2>`
 - Note: <num 1> refers to the port number for the bond (Internet sources) and <num 2> refers to the port number that the access point is connected to

5. Turn up bridge

- `sudo ifconfig <bridge> up`
- `sudo ifconfig <network interface 0> 0`
- `sudo ifconfig <network interface 1> 0`
- `sudo dhclient <bridge>`

Web Application

Follow the steps below to understand what each button on the web page does and what kind of information is displayed where.

1. To connect to the web application with another device, go to <http://<IP of ACAS System>:8000>, and ensure that the device is connected to the local network. Otherwise, the web application can be accessed by going to <http://localhost:8000>.
2. Click the "Get OvS Information" button to see which ports are active and which port is the active slave.
3. To see the upload and download speeds of the active slave, click the "Run Speed Test" button.
4. To change which active port is the active slave, enter the full name of the port (e.g. "enp1s0") in the text input at the bottom of the page and click the "Configure" button. A message below the "Configure" button shows whether the configuration was successful and which port is the active slave.
5. Click the "Get OvS Information" button to get updated information on the ports and active slave.

Appendix B: Installation Guide

Pre-requisites

- Kingdel Powerful Mini Desktop Computer
 - Ubuntu 20.02 LTS
 - Python 3.7+
 - Python3-Pip
- Raspberry Pi 4 Model B Board
 - Raspberry Pi OS Lite

Open vSwitch [26]

The following instructions are to be run in terminal on the Kingdel mini PC.

1. Install OvS

- `cd /usr/src/`
- `sudo wget https://www.openvswitch.org/releases/openvswitch-2.12.0.tar.gz`
- `sudo tar -xvf openvswitch-2.12.0.tar.gz`
- `sudo rm openvswitch-2.12.0.tar.gz`

2. Configure

- `export OVS_DIR=/usr/src/openvswitch-2.12.0`
- `cd $OVS_DIR`
- `./configure`

3. Build

- `make`
- `make check`
- `sudo make install`

4. Create OvS Database

- `sudo ovsdb-tool create /usr/local/etc/openvswitch/conf.db /usr/local/share/openvswitch/vswitch.ovsschema`
- `sudo mkdir -p /usr/local/var/log/openvswitch`
- `sudo mkdir -p /usr/local/var/run/openvswitch`

5. Start ovsdb-server

- `sudo /usr/local/share/openvswitch/scripts/ovs-ctl start`

6. Configure ovsdb-server to run on system startup

- `vim startup.sh`
 - Note: This script can be in any directory on your machine
- Add line `”sudo /usr/local/share/openvswitch/scripts/ovs-ctl start”` to file
- `chmod 770 startup.sh`
- `sudo crontab -e`
- add line `”@reboot <path to script> ”` to file
- Save changes
 - `^X`
 - `Y`
 - `enter`

NGINX Web Server [8]

The following instructions are to be run in terminal.

1. Install the NGINX web server
 - `sudo apt install nginx`
2. Edit the NGINX web server configuration settings
 - `sudo vi /etc/nginx/site-enabled/default`
3. Set the NGINX web server configuration settings

```
1 server
2 {
3     listen 80;
4
5     location /
6     {
7         proxy_pass http://127.0.0.1:8000
8         proxy_http_version 1.1;
9     }
10 }
```

4. Restart the NGINX web server
 - `sudo systemctl restart nginx`

Django Web Application

The following instructions are to be run in terminal.

1. Go to user’s home directory
 - `cd ~/`
2. Retrieve the ACAS web application from GitHub Repository
 - `git clone https://github.com/sliu20/acas-fe.git`
3. Change to the **acas-fe** directory

- cd acas-fe/
4. Install the Python 3 modules for the web application
 - pip3 install -r requirements.txt
 5. Change to the **acas-webapp** directory
 - cd acas-webapp/
 6. Run the web application in the background
 - python3 manage.py runserver &

Access Point (Raspberry Pi 4 Model B) [15]

The following instructions are to be run in terminal.

1. Install hostapd, dnsmasq, and bridge-utils
 - sudo apt install hostapd dnsmasq bridge-utils
2. Stop the hostapd and dnsmasq service since the configuration files will be changed
 - sudo systemctl stop hostapd
 - sudo systemctl stop dnsmasq
3. Unmask and enable the hostapd software
 - sudo systemctl unmask hostapd
 - sudo systemctl enable hostapd

Configure a static IP for the wlan0 interface

4. Edit the dhcpd configuration file
 - sudo vi /etc/dhcpd.conf
5. Add the dhcpd configuration settings to the end of the file (**Note:** <number> should be in the range of [2,255]. 0 and 1 should **NOT** be used since these are usually used in the default configurations of most home routers.)

```

1 interface wlan0
2 static ip_address=192.168.<number>.0/24
3 denyinterfaces eth0
4 denyinterfaces wlan0

```

Configure the DHCP server (dnsmasq)

6. Move the dnsmasq configuration file
 - sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
7. Create a new dnsmasq configuration file
 - sudo vi /etc/dnsmasq.conf
8. Set the dnsmasq configuration settings

```

1 interface=wlan0
2 dhcp-range=192.168.<number>.1,192.168.<number>.30,255.255.255.0,24h

```

Configure the host access point software (hostapd)

9. Create the hostapd configuration file

- `sudo vi /etc/hostapd/hostapd.conf`

10. Set the hostapd configuration settings

```
1 interface=wlan0
2 bridge=br0
3 hw_mode=g
4 channel=7
5 wmm_enabled=0
6 macaddr_acl=0
7 auth_algs=1
8 ignore_broadcast_ssid=0
9 wpa=2
10 wpa_key_mgmt=WPA-PSK
11 wpa_pairwise=TKIP
12 rsn_pairwise=CCMP
13 ssid=<access point name>
14 wpa_passphrase=<passphrase>
```

11. Edit the system hostapd file

- `sudo vi /etc/default/hostapd`

12. Set the location of the hostapd configuration file

```
1 DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Set up traffic forwarding

13. Edit the system configuration file

- `sudo vi /etc/sysctl.conf`

14. Set the system configuration settings

```
1 net.ipv4.ip_forward=1
```

Add a new iptables rule

15. Add IP masquerading for outbound network traffic on eth0

- `sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`

16. Save the new iptables rule

- `sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"`

17. Edit the **rc.local** file

- `sudo vi /etc/rc.local`

18. Add line above **exit 0**

```
1 iptables-restore < /etc/iptables.ipv4.nat
```

Enable internet connection

19. Add a new network bridge
 - `sudo brctl addbr br0`
20. Connect eth0 to the new bridge
 - `sudo brctl addif br0 eth0`
21. Edit the **interfaces** file
 - `sudo vi /etc/network/interfaces`
22. Add the interface settings to the end of the file

```
1 auto br0
2 iface br0 inet manual
3 bridge_ports eth0 wlan0
```

Reboot and connect to wireless access point

23. Start hostapd and dnsmasq service
 - `sudo systemctl start hostapd`
 - `sudo systemctl start dnsmasq`
24. Reboot
 - `sudo reboot`
25. Connect to wireless access point

Wi-Fi Source (Raspberry Pi 4 Model B) [23]

The following instructions are to be run in terminal.

1. Install dnsmasq
 - `sudo apt install dnsmasq`

Configure a static IP for the wlan0 interface

2. Edit the dhcpcd configuration file
 - `sudo vi /etc/dhcpcd.conf`
3. Add the dhcpcd configuration settings to the end of the file (**Note:** <number> should be in the range of [2,255]. 0 and 1 should **NOT** be used since these are usually used in the default configurations of most home routers.)

```
1 interface eth0
2 static ip_address=192.168.<number>.0/24
```

Configure the DHCP server (dnsmasq)

4. Move the dnsmasq configuration file
 - `sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig`
5. Create a new dnsmasq configuration file
 - `sudo vi /etc/dnsmasq.conf`
6. Set the dnsmasq configuration settings

```
1 interface=wlan0
2 dhcp-range=192.168.<number>.1,192.168.<number>.30,255.255.255.0,24h
```

Set up traffic forwarding

7. Edit the system configuration file
 - `sudo vi /etc/sysctl.conf`
8. Set the system configuration settings

```
1 net.ipv4.ip_forward=1
```

Add a new iptables rule

9. Add IP masquerading for outbound network traffic on eth0
 - `sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE`
10. Save the new iptables rule
 - `sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"`
11. Edit the **rc.local** file
 - `sudo vi /etc/rc.local`
12. Add line above **exit 0**

```
1 iptables-restore < /etc/iptables.ipv4.nat
```

Reboot and connect to wireless internet source

13. Start dnsmasq service
 - `sudo systemctl start dnsmasq`
14. Reboot
 - `sudo reboot`
15. Connect to wireless internet source

Appendix C: Hardware Parts and Specifications

Router: Kingdel Powerful Mini Desktop Computer

- CPU: Intel i5-3317U, Dual Core, Quad Thread; Base Frequency: 1.7GHz, Max. Turbo Frequency: 2.6GHz, 3MB Smart Cache
- 8GB DDR3L RAM
- 128GB mSATA SSD(Solid State Disk)
- Graphics: Intel HD Graphics 4000: Base Frequency: 350 MHz; Max Dynamic Frequency: 1.05 GHz
- 2 NICs Intel 82574L Gigabyte LAN Port
- 4 COM RS232 ports; HD port; VGA ports
- 4 USB 3.0 ports; 4 USB 2.0 ports
- Wireless Type: 802.11bgn
- Operating System: Ubuntu 20.02 LTS

Access Point and Wi-Fi Source: Raspberry Pi 4 Model B

- CPU: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 4GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Micro-SD card slot for loading operating system and data storage
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports
- 2 micro-HDMI ports (up to 4kp60 supported)
- Operating System: Raspberry Pi OS Lite

Appendix D: Source Code

The following is code from the views.py file of our Django web application. To see all of the code for the web application, please visit <https://github.com/sliu20/acas-fe>.

```
1 from django.shortcuts import render
2
3
4 import speedtest
5 import subprocess
6 import re
7
8
9 st = speedtest.Speedtest()
10
11
12 # Function: execute(str)
13 # Description: Returns a string of the output of the executed command
14 def execute(command):
15     command = command.split(' ')
16     output = subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
17     result = output.stdout.decode('utf-8')
18     if len(result) == 0:
19         result = output.stderr.decode('utf-8')
20     return result
21
22
23 # Function: getUpload():
24 # Description: Returns a float of the upload bandwidth (bps)
25 def getUploadSpeed():
26     return st.upload()
27
28
29 # Function: getDownload():
30 # Description: Returns a float of the download bandwidth (bps)
31 def getDownloadSpeed():
32     return st.download()
33
34
35 # Function: convert2Mega(float)
36 # Description: Returns a float of the number given converted with the metric prefix Mega
37 def convert2Mega(x):
38     return x / (10**6)
39
40
41 # Function: getBondPortStatus(str)
42 # Description: Returns a list of tuples for the ports used and the status of each port in the
43     bond
44 def getBondPortStatus(bondCommandOutput):
45     status = re.findall(r"\nslave (.*): (.*)\n", bondCommandOutput)
46
47     if len(status) == 0:
48         return None
49
50     for i in range(len(status)):
51         if status[i][1] == "enabled":
```



```

51     status[i] = (status[i][0], "available")
52     else:
53         status[i] = (status[i][0], "unavailable")
54
55     return status
56
57
58 # Function: getActiveSlave(str)
59 # Description: Returns a list of the active slave port in the bond
60 def getActiveSlave(bondCommandOutput):
61     active = re.findall(r"\nslave (.*) : enabled\s+active slave\n", bondCommandOutput)
62     if len(active) == 0:
63         active = None
64     else:
65         active = active[0]
66
67     return active
68
69
70 # Function: getConfigureActiveSlave(str, str)
71 # Description: Returns a string of the success message of the configuration of the active slave
72 def getConfigureActiveSlaveMessage(configureCommandOutput, port):
73     message = re.findall(r"(.*)\n", configureCommandOutput)
74     message = message[0]
75
76     if message == "done":
77         message = "Success: The active slave is now " + port + "."
78     elif message == "no change":
79         message = "Warning: Port " + port + " is already the active slave."
80     elif message == "no such slave":
81         message = "Error: There was no such port " + port + " used in the bond. Please select an
            available port that is used in the bond."
82     elif message == "cannot make disabled slave active":
83         message = "Error: Port " + port + " is unavailable. Please select an available port."
84     else:
85         message = "Error: Something went wrong. Try again or contact system admin to resolve
            persisting issues."
86
87     return message
88
89
90 # GLOBAL VARIABLES
91 uploadSpeedbps = None
92 downloadSpeedbps = None
93 uploadSpeedMbps = None
94 downloadSpeedMbps = None
95 bondPortStatus = None
96 activeSlave = None
97 configureActiveSlaveMessage = None
98
99
100 # Function: statsView()
101 # Description: Returns the components for the statistics page
102 def statsView(request):
103     global uploadSpeedbps
104     global downloadSpeedbps
105     global uploadSpeedMbps
106     global downloadSpeedMbps
107     global bondPortStatus
108     global activeSlave
109     global configureActiveSlaveMessage
110
111     configureActiveSlaveMessage = None
112
113     if request.POST.get("run_speed_test"):
114         upload = getUploadSpeed()
115         download = getDownloadSpeed()
116         uploadSpeedbps = "{:.0f}".format(upload)

```

```

117     downloadSpeedbps = "{:.0f}".format(download)
118     uploadSpeedMbps = "{:.2f}".format(convert2Mega(upload))
119     downloadSpeedMbps = "{:.2f}".format(convert2Mega(download))
120     elif request.POST.get("get_ovs_information"):
121         command = "sudo ovs-appctl bond/show bond0"
122         commandResult = execute(command)
123         bondPortStatus = getBondPortStatus(commandResult)
124         activeSlave = getActiveSlave(commandResult)
125     elif request.POST.get("configure_active_slave"):
126         port = request.POST.get("port")
127         command = "sudo ovs-appctl bond/set-active-slave bond0 " + port
128         commandResult = execute(command)
129         configureActiveSlaveMessage = getConfigActiveSlaveMessage(commandResult, port)
130
131     context = {
132         'uploadSpeedbps': uploadSpeedbps,
133         'downloadSpeedbps': downloadSpeedbps,
134         'uploadSpeedMbps': uploadSpeedMbps,
135         'downloadSpeedMbps': downloadSpeedMbps,
136         'bondPortStatus': bondPortStatus,
137         'activeSlave': activeSlave,
138         'configureActiveSlaveMessage': configureActiveSlaveMessage,
139     }
140
141     return render(request, "statistics.html", context)

```

Appendix E: Senior Design Presentation

Presentation slides from the 2020 Senior Design Conference hosted by the Santa Clara University School of Engineering are included in this section.



Senior Design Conference

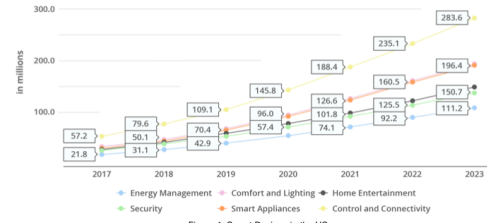
ACAS: Always Connected, Always Secure

Blaise Aranador, Antonio Gigliotti, Shining Liu
Advisor: Dr. Behnam Dezfouli

May 2020

Problem

- Safety does not come easy
 - **Home burglary every 22 seconds!**^[1]
- Consumers turning to smart security devices
 - 40+ million such devices in the US^[2]
- These devices rely on Internet connection
 - **What happens without that connection?**



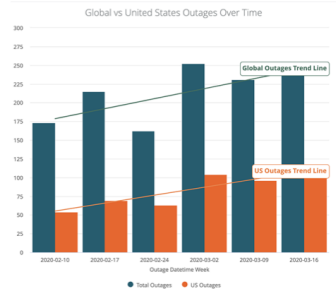
Existing Solutions

- **Load balancers**
 - Additional equipment (costly)
 - High power consumption
 - Not fully programmable
 - Not extensible → limited ports
- **Cellular data**
 - Additional equipment (costly)
 - Subscription required



Motivation


- Rise in Internet usage and outages over last few months
- Outages can happen anytime
 - Power loss
 - ISP failure
 - Bad wires
 - Extreme weather
 - Extreme usage
 - Hardware issues




ACAS: Always Connected, Always Secure

Solution


Features




Open Source




Fast & Efficient



Affordable



Reliable Backup



Public Wi-Fi Availability

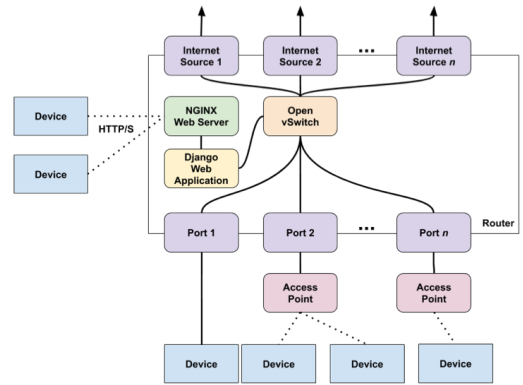
System Overview



- Smart connection **monitoring and recovery** agent
 - Monitor network connectivity
 - Switch to alternative connections
 - Application for user interaction
- Components
 - Linux
 - Open Virtual Switch (OvS)
 - OpenFlow Protocol
 - NGINX web server
 - Web application (Django)



System Architecture



Hardware



Kingdel Mini PC



- Intel Core i5-3437U @ 1.90GHz processor
- 4GB RAM
- Gigabit Ethernet
- Wi-Fi and Bluetooth capabilities
- OS: Linux (Ubuntu)

Raspberry Pi 4



- Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz processor
- 4GB RAM
- Gigabit Ethernet
- Wi-Fi and Bluetooth capabilities

Open Virtual Switch



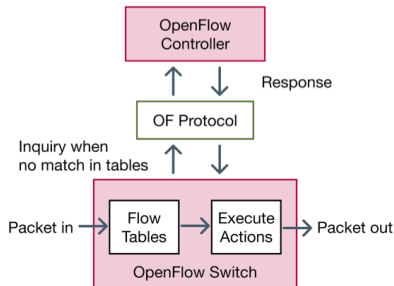
- Production quality, virtual switch
- Used to connect VMs in large scale cloud computing networks
- **Bridge**
 - Virtual network bridge
 - Creates single aggregate connection between interfaces connected to bridge
- **Bond**
 - Allows 2 or more interfaces to share network traffic and act as a single port
 - Active backup



OpenFlow Protocol



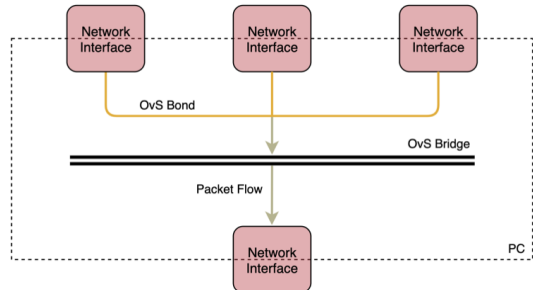
- Protocol that controls packet flow
- Connected to OvS bridge and configures the switch



Software Architecture



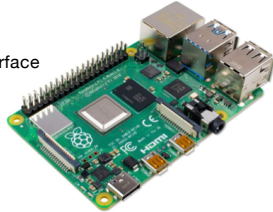
- We employ bonding to determine which network interface to use at any given time.



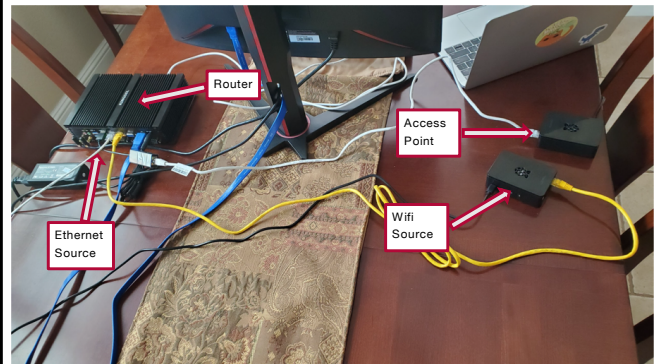
Access Point



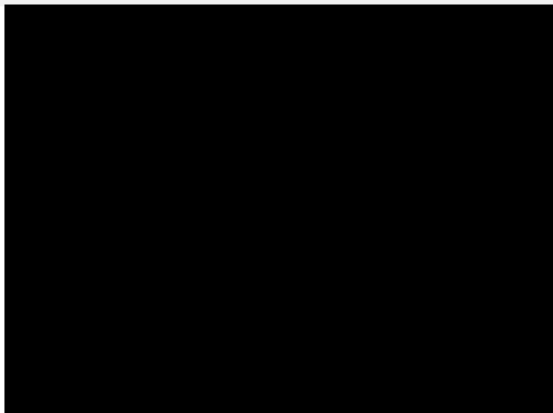
- Raspberry Pi 4 used to build access point functionality
 - **hostAPD**
 - Access point daemon software
 - Provide access point capabilities
 - **DNSmasq**
 - Allocates IP addresses to connected devices
 - DNS caching
 - Router advertisement
 - Network boot features
 - **Bridge-Utils**
 - Network traffic forwarding
 - Wireless interface wired interface



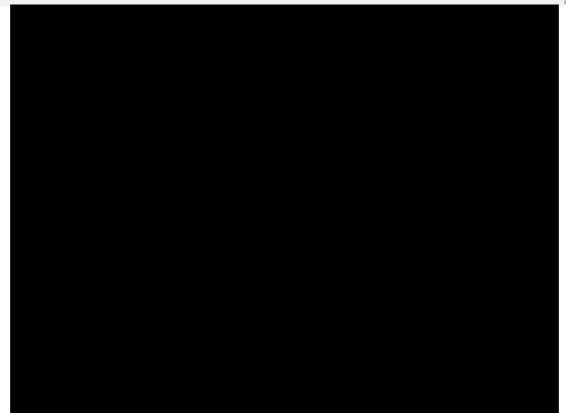
System Components



OvS Configuration



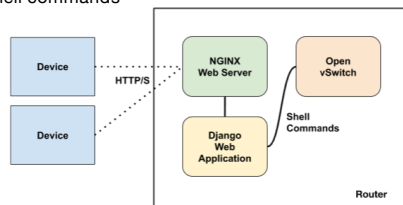
System Demo



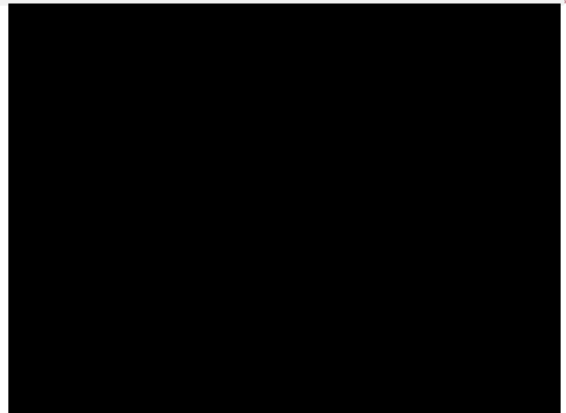
Web Application (Back-End)

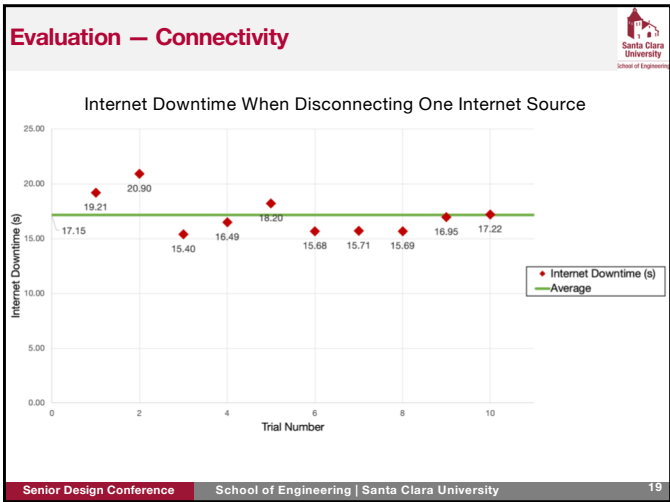


- NGINX web server
 - Allows user devices (phone, laptop, etc.) to connect to the web application
 - HTTP/S communication
- Django web application
 - Python
 - SpeedTest.net
 - Shell commands



Web Application (Front-End Demo/Screenshot)






ACAS: Always Connected, Always Secure


Remarks

Senior Design Conference | School of Engineering | Santa Clara University | 20


Societal Issues



Manufacturability



Open Source




Health & Safety


Senior Design Conference | School of Engineering | Santa Clara University | 21

- ### Obstacles Encountered
- Unforeseen circumstances
 - Lack of documentation/resources
 - Working remotely
 - Shipping equipment
 - Design changes
 - DPDK compatibility
 - Wi-Fi connectivity
 - Mobile to web application
 - Time management and constraints
 - Additional equipment required
- Senior Design Conference | School of Engineering | Santa Clara University | 22


Lessons Learned




Consistent Meetings




Clarify Requirements



Start Early



Research First



Be Prepared

Senior Design Conference | School of Engineering | Santa Clara University | 23

- ### Conclusion
- Reliable internet is important
 - Smart connection monitoring and recovery system
 - Open-source software
 - Commodity hardware
 - Reliable backup
 - Configurable
 - Affordable
 - Many applications beyond just smart home security
- Senior Design Conference | School of Engineering | Santa Clara University | 24

Questions?

References



- [1] "Burglary", FBI, September 2018. [Online]. Available: <https://ucr.fbi.gov/crime-in-the-u.s/2017/crime-in-the-u.s.-2017/topic-pages/burglary?tkid=62548>. [Accessed October 7, 2019]
- [2] J. Bustamante, "Smart Home IoT Statistics", iPropertyManagement, [Online]. Available: <https://propertymanagement.com/research/iot-statistics>. [Accessed December 8, 2019]
- [3] C. Reynolds, "New Global Internet Outages Map: "Concerning" Rise in ISP Outages", Computer Business Review, [Online]. Available: <https://www.cbonline.com/news/global-internet-outages-map/#>. [Accessed May 15, 2020]
- [4] "What are Xfinity WiFi Hotspots and how do I connect?", Xfinity Mobile, [Online]. Available: <https://www.xfinity.com/mobile/support/article/xfinity-mobile-wifi-hotspots>. [Accessed May 15, 2020]

References

- [1] Shadi Abou-Zahra. *Colors with Good Contrast*. URL: <https://www.w3.org/WAI/perspective-videos/contrast/>. [Accessed December 30, 2019].
- [2] Alarms.org. *Burglary Statistics: The Hard Numbers*. URL: <https://www.alarms.org/burglary-statistics/>. [Accessed October 8, 2019].
- [3] Brad Berman and Jennifer Pattison Tuohy. *SimpliSafe Home Security System Review and Prices*. URL: <https://www.usnews.com/360-reviews/home-security/simplisafe>. [Accessed May 27, 2020].
- [4] Jaleesa Bustamante. *Smart Home IoT Statistics*. URL: <https://ipropertymanagement.com/research/iot-statistics>. [Accessed December 8, 2019].
- [5] Matt Conran. *OPEN VSWITCH (OVS) BASICS*. URL: <https://network-insight.net/2015/11/open-vswitch-ovs-basics/>. [Accessed November 12, 2020].
- [6] The Mozilla Developers. *Django Introduction*. URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>. [Accessed November 12, 2019].
- [7] Jim Duffy. *FAQ: What is OpenFlow and why is it needed?* URL: <https://www.networkworld.com/article/2202144/data-center-faq-what-is-openflow-and-why-is-it-needed.html>. [Accessed May 12, 2020].
- [8] Justin Ellingwood and Kathleen Juell. *How To Install Nginx on Ubuntu 18.04*. URL: <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04>. [Accessed November 30, 2019].
- [9] FBI. *Burglary*. URL: <https://ucr.fbi.gov/crime-in-the-u.s/2017/crime-in-the-u.s.-2017/topic-pages/burglary?kbid=62548>. [Accessed October 7, 2019].
- [10] F5 Glossary. *Load Balancer*. URL: <https://www.f5.com/services/resources/glossary/load-balancer>. [Accessed May 28, 2020].
- [11] Kris Gunnars. *How to Optimize Web Fonts for Screen Readability*. URL: <https://searchfacts.com/web-font-screen-readability/>. [Accessed December 30, 2019].
- [12] *How to configure and test LACP bonding with Open vSwitch DPDK?* URL: <https://access.redhat.com/solutions/3417681>. [Accessed April 30, 2020].
- [13] imperva. *Load Balancer Hardware*. URL: <https://www.imperva.com/learn/availability/hardware-load-balancer-hld/>. [Accessed May 28, 2020].
- [14] Alexandra Leslie. *NGINX vs. Apache (Pro/Con Review, Uses, Hosting for Each)*. URL: <https://www.hostingadvice.com/how-to/nginx-vs-apache/>. [Accessed November 9, 2019].
- [15] Stephen Lovely. *How To Use Your Raspberry Pi As A Wireless Access Point*. URL: <https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/>. [Accessed April 28, 2020].
- [16] Matt 'Sivel' Martz. *speedtest-cli*. URL: <https://pypi.org/project/speedtest-cli/>. [Accessed November 28, 2019].
- [17] *Networking Benchmarks*. URL: <https://www.pidramble.com/wiki/benchmarks/networking>. [Accessed December 20, 2019].
- [18] Neil Patel. *12 Essential Tips to Picking a Website Color Scheme*. URL: <https://neilpatel.com/blog/website-color-scheme/>. [Accessed December 30, 2019].

- [19] Red Hat Customer Portal. *APPENDIX G. OPEN VSWITCH BONDING OPTIONS*. URL: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/8/html/director_installation_and_usage/appe-bonding_options. [Accessed April 28, 2020].
- [20] *Python - Modules*. URL: https://www.tutorialspoint.com/python/python_modules.htm#:~:text=A%20module%20is%20a%20Python,can%20also%20include%20runnable%20code.. [Accessed November 28, 2019].
- [21] Mindfire Solutions. *Python: 7 Important Reasons Why You Should Use Python*. URL: <https://medium.com/@mindfiresolutions.usa/python-7-important-reasons-why-you-should-use-python-5801a98a0d0b#:~:text=Python%20is%20a%20general%20purpose%20programming%20language,facilitate%20data%20analysis%20and%20visualization..> [Accessed November 28, 2019].
- [22] Sparklin. *Top 5 UI fonts for website mobile apps*. URL: <https://medium.muz.li/top-5-ui-fonts-for-website-mobile-apps-d78829e58f7e>. [Accessed December 30, 2019].
- [23] SurferTim. *Wifi Ethernet shared through LAN*. URL: <https://www.raspberrypi.org/forums/viewtopic.php?t=223295>. [Accessed April 28, 2020].
- [24] Open vSwitch. *Basic Configuration*. URL: <http://docs.openvswitch.org/en/latest/faq/configuration/>. [Accessed April 1, 2020].
- [25] Open vSwitch. *Bonding*. URL: <http://docs.openvswitch.org/en/latest/topics/bonding/>. [Accessed April 28, 2020].
- [26] Open vSwitch. *Open vSwitch on Linux, FreeBSD and NetBSD*. URL: <http://docs.openvswitch.org/en/latest/intro/install/general/>. [Accessed on March 28, 2020].
- [27] Open vSwitch. *Using OpenFlow*. URL: <http://docs.openvswitch.org/en/latest/faq/openflow/>. [Accessed April 27, 2020].
- [28] Open vSwitch. *Why Open vSwitch?* URL: <http://docs.openvswitch.org/en/latest/intro/why-ovs/>. [Accessed May 12, 2020].
- [29] WebAIM. *Contrast and Color Accessibility: Understanding WCAG 2 Contrast and Color Requirements*. URL: <https://webaim.org/articles/contrast/#ratio>. [Accessed December 30, 2019].
- [30] *What Is an Application Server vs. a Web Server?* URL: <https://www.nginx.com/resources/glossary/application-server-vs-web-server/>. [Accessed November 13, 2019].