Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

Miro Lahdenmäki

Software Visualization for Teaching Network Protocols

Master's Thesis
Espoo, June 2 2010

Supervisor: Professor Antti Ylä-Jääski, Aalto University
Instructors: Sanna Suoranta and Jan Lönnberg, Aalto University

Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Miro Lahdenmäki	
Title of thesis:	Software Visualization for Teaching Network Protocols	
Date:	June 2 2010	**Pages:** 10 + 63
Professorship:	Data Communications Software	**Code:** T-110
Supervisor:	Professor Antti Ylä-Jääski	
Instructors:	Sanna Suoranta and Jan Lönnberg	

The objective of this thesis is to determine how to create an interactive system to aid in teaching data communications protocols and enhance learning by means of software visualization. The visualization needs to support varied network protocols, and adding new ones to the system should be easy. The system should support lecture demonstrations, and learning by oneself should also be possible, taking into account that students today are accustomed to extensive interaction and instant gratification.

A brief survey is made of the field of software visualization. Several systems are looked at and their suitability, concerning the objectives defined in this thesis, is reviewed. Host Identity Protocol is used as an example for visualization as it is a fairly simple protocol but can involve several communicating parties. The Matrix framework is chosen as the platform to build upon. It provides much of the features that are seen as important in solving the problems presented in this thesis. It has been proven successful for teaching data structures and algorithms. Alternative solutions and approaches are assessed. Example scenarios demonstrate the proposed solution in the form of a cognitive walkthrough. The design is assessed utilizing Price's taxonomy and usability heuristics. Future work involves implementing the solution and assessing the system in actual use on the coming data communications courses.

Keywords:	Learning, teaching, protocols, software, visualization, interactive, Matrix, Host Identity Protocol
Language:	English

Aalto University
School of Science
and Technology

Aalto-yliopisto
Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Miro Lahdenmäki

Työn nimi:
Ohjelmistohavainnollistus tietoliikenneprotokollien opetukseen

Päiväys: 2. kesäkuuta 2010 **Sivumäärä:** 10 + 63
Professuuri: Tietoliikenneohjelmistot **Koodi:** T-110
Työn valvoja: Professori Antti Ylä-Jääski
Työn ohjaajat: Sanna Suoranta ja Jan Lönnberg

Tämän diplomityön tavoitteena on määritellä, miten luoda tietoliikenneprotokollien opetuksen tueksi ja oppimisen helpottamiseksi vuorovaikutteinen ohjelmisto ohjelmistopohjaisen havainnollistamisen keinoin. Havainnollistuksen on tuettava erilaisia tietoliikenneprotokollia, ja uusien protokollien lisäämisen ohjelmistoon tulisi olla helppoa. Myös esitysten luomisen luentoa varten pitäisi olla helppoa. Ratkaisun tulisi tukea itsenäistä opiskelua ottaen huomioon, että oppilaat ovat tottuneet laajaan vuorovaikutteisuuteen sekä välittömään mielihyvään.

Työssä luodaan lyhyt katsaus ohjelmistopohjaisen havainnollistamisen kentälle. Ohjelmistoja käydään läpi, ja niiden soveltavuutta arvioidaan esitettyjen tavoitteiden saavuttamiseksi. Host Identity Protocol:aa käytetään esimerkkinä, sillä se on kohtuullisen yksinkertainen tietoliikenneprotokolla, jonka osallisena voi olla useitakin osapuolia. Alustaksi ratkaisulle valitaan Matrix-sovelluskehys. Se tarjoaa monia ominaisuuksia, jotka on nähty tärkeinä työssä esitettyjen ongelmien ratkaisemiseksi. Sitä on myös käytetty menestyksekkäästi tietorakenteiden ja algoritmien opetuksessa. Vaihtoehtoisia lähestymistapoja ja ratkaisuja havainnollistuksiin arvioidaan. Ratkaisua esitellään kognitiivisen läpikäynnin avulla, ja arvioidaan Pricen taksonomiaa hyödyntäen sekä käytettävyysheuristiikkojen avulla. Jatkokehittelyyn jää itse toteutus ja systeemin käytön arviointi tulevalla tietoliikenteen kurssilla.

Avainsanat: Oppiminen, opettaminen, protokolla, vuorovaikutteinen, ohjelmistopohjainen, havainnollistus, Matrix, HIP
Kieli: englanti

Aalto-yliopisto
Teknillinen korkeakoulu

Acknowledgements

I started working on this thesis already at the end of 2007. Progress was slow at first as I concentrated more on my other duties. At the end of April in 2008 my sweet baby girl was born and that brought up with it some new priorities to my life. Now that the work is finally finished, I want to thank everyone involved for making it possible.

I thank my instructors Sanna Suoranta and Jan Lönnberg for their valuable guidance and support. Sanna acted as my instructor all the way from the beginning. She was patient and supportive with me, and gave excellent guidance. Jan became my second instructor halfway through the process and gave invaluable support and was an excellent source of information, especially concerning software visualization. I want to thank my supervisor, professor Antti Ylä-Jääski, for allowing me to work on my thesis so long and believing in me.

I want to thank my parents for all the support they have given me along the years, supporting my studies, and being great grandparents. Last and, certainly, not least, I want to thank the love of my life, Saarajohanna, for supporting me, enduring with me, and making sacrifices for me.

Espoo June 2 2010

Miro Lahdenmäki

Abbreviations and Acronyms

3G	3rd Generation
ACK	Acknowledgement
ADT	Abstract data type
AES	Advanced Encryption Standard
D-H	Diffie-Hellman
ESP	Encapsulating Security Payload
FDT	Fundamental data type
HI	Host Identity
HIP	Host Identity Protocol
HIT	Host Identity Tag
HMAC	Hash-based Message Authentication Code
IKE	Internet Key Exchange Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security suite
LAN	Local Area Network
MSC	Message Sequence Chart
NAT	Network Address Translation
ORCHID	Overlay Routable Cryptographic Hash Identifiers
RFC	Request for Comments
SHA	Secure Hash Standard
sig	Signature
SPI	Security Parameter Index
TCP	Transmission Control Protocol

Definitions

Matrix	Portable algorithm simulation framework
MatrixPro	Demostration tool for Matrix

Contents

List of Figures

Chapter 1

Introduction

The Internet has changed our everyday lives. We have grown accustomed to having everything available on the Internet, be it news or publications, map services, timetables, translation services, social networking, personal communication, forums, encyclopedias, entertainment, reservation services, banking, shopping, or a calendar. We are also starting to want the same services available to us wherever we go and whatever the time using the network devices of our choice. Since the Internet has such a profound effect on our lives these days, developers need to understand how it works and how the services above are created and made mobile. To understand how the Internet works, the developer needs to understand the lower level services that enable the Internet, and everything built on top of it. The services enabling the Internet are provided in the form of data communications protocols, so understanding them is the key.

Students have grown up in a modern digitalized society. People spend a lot of time playing digital games, online and instant messaging. For example, over 80% of 6 to 24 year old persons could be counted as heavy gamers, and 84% of 16-24 year old persons used the PC for instant messaging in the United Kingdom in a study published in 2005 [33]. Figure 1.1 shows the percentage in different age groups in the United Kingdom who play games. Junco and Mastrodicasa [15] suggest that Net Generation students expect interaction and constant feedback.

Many data communications protocols are complicated and explaining them can be demanding. Data communications protocols in general have parties communicating to each other trying to reach some common goal, such as setting up a trustworthy connection for sending messages to each other over the Internet. With the growing number of services and applications using the Internet, also the network topology of Internet has become more complex. Among other things, this is due to the demands of mobility and data security.

The purpose of this thesis work is to design a system for illustrating data communications protocols interactively. This is done by extending the Matrix platform. Matrix has been used successfully in illustrating how algorithms work. Korhonen [19] reports

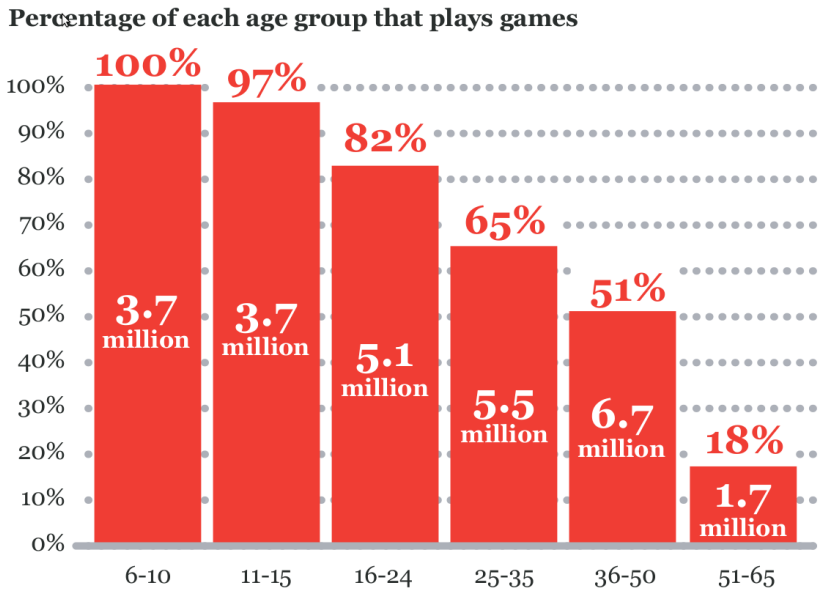

Figure 1.1: Percentage of each age group in UK who play games [33].

that using Matrix as a tool for teaching and assignments has resulted in as good learning results as tutorial teaching. In addition, it can ease the work of the teacher and the assistants, and offers instant feedback to the students.

The goal is that, with the designed system, a teacher can prepare a demo for a lecture illustrating how a protocol works, such as what messages are exchanged, what information is exchanged, in what state the communicating parties are, what happens in different error situations, how the state of the network affects the operation of the protocol, or even how a certain information security breach is enabled or averted by the protocol. Using TRAKLA2, built on Matrix, exercises can be created for the implemented protocols. The system will check the answers automatically and it gives feedback instantly. This is a great benefit for the student since the student has the exercise still in short-term memory and thus learning from errors is easier. The student can also study data communications protocols by interactively simulating them, and, at the same time, study the information exchange and read more about the phases and information elements involved.

The purpose of the system is to support, especially, basic teaching of data communications. There are many applications that visualize and analyze Internet traffic, but these often concentrate on traffic analysis and are oriented toward administration purposes and people who already know their way around network protocols. The novelty of the system described in this thesis is the ease of creating demos for lectures, the automatic generation and checking of exercises, and the possibility of interactively simulating a data communications protocol while learning more about it. This functionality comes from building on top of the Matrix platform.

1.1 Problem Statement

The scope and definition of the research problem are stated in this chapter.

1. Demonstrating Internet principles and protocols is difficult because there is such a diversified set of them.
2. Protocol visualizations are often confined to previously determined protocols and message exchange flows.
3. There may be multiple parties involved in message exchange. Depicting this can be hard.
4. Courses often have a massive amount of students and few assistants. There is little possibility for personal guidance, and checking exercises for all the students is a lot of monotonous work.
5. In some part connected with the previous, getting feedback may be impossible or students get it so late that learning is hindered.
6. Solutions often do not support trying out for yourself.

From the previous statements, I conclude that the objective of this thesis is to determine how to create an interactive system to aid in teaching protocols and enhance learning by means of visualizing. I want to tackle the problems mentioned above in my solution.

1.2 Example scenario

Here I present two imaginary example scenarios. In the first scenario, a teacher is using the visualization presented in this thesis on a lecture to present different aspects of Host Identity Protocol. In the second scenario a student is using a system implementing the visualization to study, among other things, connection establishment for HIP.

1.2.1 Scenario A: Lecture

Host Identity Protocol (HIP) is the day's lecture topic in an advanced course on computer networks. The teacher first gives an introduction and some background information on HIP utilizing a set of slides. While describing the protocol, he displays the HIP handshake message exchange on the classroom screen, step by step. The final step is visible in Figure 1.2.

The teacher proceeds by laying out a situation where Alice has established a HIP association through 3G and wants to use a faster wireless local area network which happens

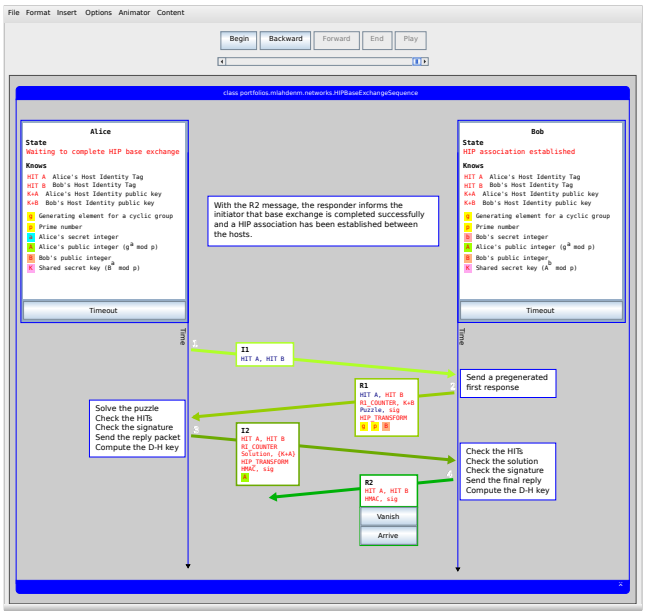

Figure 1.2: HIP connection establishment.

to be available. He shows the required message exchange in HIP. In the following figure 1.3, Alice has just sent the first UPDATE message. The teacher has clicked on the LOCATOR parameter of the UPDATE message. This has opened a box showing the sub parameters NEW_LOCATOR and ORIG_LOCATOR. Clicking the LOCATOR has also brought up new information in the upper information box. Then the teacher has clicked on the NEW_LOCATOR sub-parameter, which has opened yet another box containing Alice's 2nd address, 2nd inbound SPI, and a boolean value indicating it as the preferred locator. When the students ask questions, the teacher can easily, through the visualization, show the different elements involved, go back and forth in the message exchange, and even take a different route.

1.2.2 Scenario B: Studying individually

After being on a lecture on Host Identity Protocol the same day, Alice is at home. One thing about the keys used in HIP base exchange was left unclear to her during a demo on the lecture, and she wants to get back to it. She points her browser to the course web page, and in a few moments she is presented with a similar view as the one the teacher had in his demo on her own screen. She selects the HIP base exchange from the menu and is presented with the view displayed in Figure 1.4.

She starts simulating the base exchange by clicking the available buttons which control the sending of the messages and simulate the network. She also clicks on elements, like Shared secret key, to learn about them. After going through the base exchange she checks a couple more HIP demos from the menu to verify she has really understood

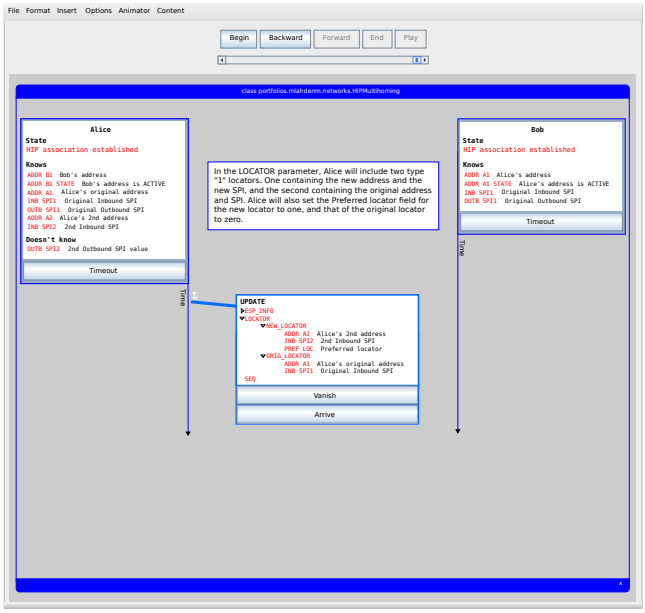

Figure 1.3: HIP multihoming scenario.

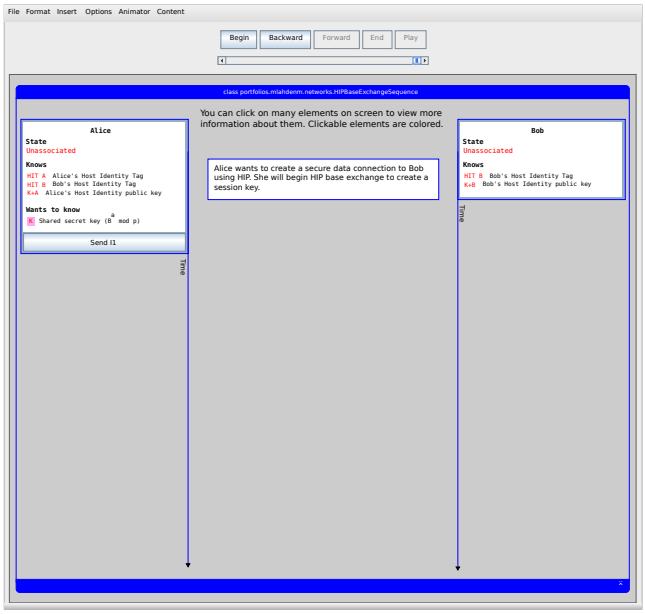

Figure 1.4: HIP connection establishment welcome screen.

how the keys work. Later she discovers that she has no problems with the course homework.

1.3 Research methodology

This thesis takes a look at existing solutions for visualizing networks and protocols, and for teaching how the Internet works. Different ideas from these sources and solutions are assessed in light of addressing the problems which this thesis tries to solve. Different solutions are weighed against each other. The best ideas are incorporated together in one design which is presented in this thesis. The design is presented in detail in a form of cognitive walkthrough, and then evaluated utilizing a taxonomy of software visualization and heuristic evaluation.

1.4 Organization of the thesis

The rest of this thesis is organized as follows: Chapter 2 gives an introduction to the protocols that are used as examples in this thesis. Recent developments in the field of software visualization and especially network and protocol visualization are reviewed in Chapter 3. Chapter 4 handles the Matrix platform and associated software. Chapter 5 presents the architecture and design of the system. The design is evaluated in Chapter 6. Chapter 7 concludes this thesis by discussing future use and development.

Chapter 2

Introduction to protocols

In computer science, a protocol defines how computers communicate with each other across a network. It is a set of rules that define the messages that can be used (the syntax), their meaning (semantics), and when to send them (synchronization). A communications protocol defines rules for sending information across a communications channel. As stated before, protocols provide services, and they work on top of each other. E.g. the Internet Protocol [31] provides the service of attempting to deliver small chunks of data across a heterogeneous network, and Transmission Control Protocol (TCP) utilizes this service to provide its own service, providing a reliable bidirectional data flow across a heterogeneous network.

Protocols may be viewed working on different layers. A protocol on one layer uses the services provided by the protocols a layer below to provide services for the protocols working a layer above. The layers of the Internet network can be named the Physical layer, the Link layer, the Network layer, the Transport layer, and the Application layer. Internet Protocol is situated on the Network layer, and the Transmission Control Protocol on the Transport layer. The Application layer has e.g. the Domain Name Resolution (DNS) protocol providing services for naming systems connected to the Internet, the Hypertext Transfer Protocol (HTTP) providing services for the World Wide Web (WWW), and Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), and Internet Message Access Protocol (IMAP) which provide email services.

Internet has been developed at a time when everyone in the network trusted everyone else, and data security was unnecessary. Today, terminals connected to the Internet can also be easily moved around. So, data communications protocols have to be continually developed to rise, among other things, to the challenges brought by data security issues and mobility. One proposed solution is the Host Identity Protocol (HIP). In the layer model, it can be placed somewhere between the Network and the Transport layers. I will next discuss two Internet protocols, TCP and HIP, which will be later used as example protocols in the visualizations presented in this thesis.

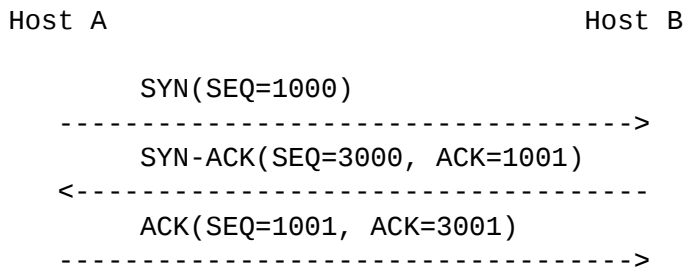

Figure 2.1: An example diagram of TCP handshake

2.1 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCPv4 [32]) provides the service of reliably delivering data packets over a heterogeneous network. TCP can recover from data that is damaged, lost, duplicated, or delivered out of order by the underlying communications system. This is achieved by attaching a sequence number to the octets of data to be delivered, and adding a checksum to each data segment.

To make it possible for multiple simultaneous processes on one host to use TCP, additional addresses called ports are used to route information. A port will tell to which process, or service, a message is meant inside a host. An IP address and a port together form a socket. To enable reliability and flow control mechanisms, TCP instances must maintain status information. This information includes a pair of sockets, sequence numbers, and window sizes for both the sender and the receiver. Together these form a connection.

Since both the hosts and the Internet are unreliable, a handshake mechanism is used to establish a connection between hosts in the network to avoid erroneous initializations of connections. The handshake mechanism uses sequence numbers which both ends randomly initialize as a measure to protect against TCP sequence prediction attacks in which an attacker may counterfeit packets. The sequence number is used for synchronization and acknowledgements. Both ends state in each TCP message what sequence number they are expecting next. Several sequences of data can be acknowledged with a single message, if all the pieces in between have arrived. Figure 2.1 shows an example diagram of the handshake. This sort of Message Sequence Chart (MSC) [12] diagram is commonly used in presenting message exchange in protocol standard definitions.

2.1.1 Flow control

TCP implements the Positive Acknowledgment with Re-Transmission (PAR) scheme defined in [32]. In this scheme, a packet is sent over and over again until the sending end receives a positive acknowledgement from the receiving end, or until a defined time period, called timeout, expires. Another name for this scheme is stop-and-wait

since the sender waits for an acknowledgement or timeout before sending the next packet or the same packet again. A receive window parameter can be used to allow more octets in several packets to be in transit simultaneously.

Sliding Window Protocol [13] is a bi-directional data transmission control method used in TCP. Its purpose is to increase throughput. It is used to keep track of sent and received packets and acknowledgements.

In the Cumulative Acknowledgement scheme, used in the TCP Sliding Window Protocol [13], the sender sends a certain number of packets (corresponding to the send window size) and then waits for an acknowledgement from the receiver. As the sender receives an acknowledgement of a packet it will send the next packet waiting to be sent. If the sender times out, it will send all the packets again starting from the first one which was not acknowledged.

The receiver, on the other hand, only accepts packets one by one and in the right order. If a packet whose sequence number is not the anticipated one arrives at the receiver, the packet is disregarded. This can happen if a packet is lost or packets arrive in the wrong order. Another name for this scheme is Go-back-n, since in case of errors, the sender will resend n packets with n being the size of the send window.

Another variant for the Sliding Window Protocol in TCP is Selective repeat defined in [23]. In the positive Selective Acknowledgement scheme (SACK) the receiving end explicitly states which messages were received, and the sender may utilize this knowledge to resend only the missing octets.

2.1.2 Congestion management

To avoid network congestion, additional congestion control algorithms are defined in RFC2581 [1] TCP Congestion Control. An implementation may follow a more conservative policy than the algorithms allow. Slow Start and Congestion Avoidance algorithms set two additional mandatory state variables: congestion window (cwnd), and receiver's advertised window (rwnd). The congestion window parameter sets a limit to the amount of data the sender can transmit before receiving an acknowledgement. The receiver's advertised window value comes from the receiving end, and the maximum amount of data in transit at the same time is limited by the minimum of these two values. Normally, these values are the same.

When there is network congestion, packets will not get through. The hosts can detect this as they do not receive acknowledgements. When this happens, the hosts lower the congestion window parameter. To avoid network congestion again, the slow start policy is used when starting to send packets. The state of the slow start threshold (sstresh) state variable determines whether slow start or congestion avoidance is used. TCP will exponentially increase the size of the send window from a small value when starting a new transmission or if congestion is detected. At other times, a congestion

avoidance scheme is used in which the send window is steadily increased.

There are different algorithms for recovering quickly when data transmission is interfered with. When an out-of-order segment arrives, the receiver should send the previous acknowledgement (ACK) again. To the sender, receiving a duplicate ACK indicates a dropped segment, re-ordering of data segments by the network, or replication of ACK's or data segments by the network. Also, if an incoming segment fills in a gap, the receiver should send another ACK. The previous measures make it possible to use a retransmission timeout, a fast transmit, or another loss recovery algorithm, such as New Reno, Vegas, BIC, CUBIC, or Compound TCP. These use loss-based or delay-based congestion control, some with a more aggressive, some with a less aggressive rate control. The different algorithms work best in different network types.

2.2 Host Identity Protocol (HIP)

A traditional IP address is an identifier of a network interface and a locator of its place in the network topology. TCP, for its part, creates a connection between two hosts through a network but does not guarantee the other peer is who it claims to be. Neither does it guarantee that no one reads or alters the messages and their checksums on their way through the network. The Host Identity Protocol separates identifier from locator at the network layer of the TCP/IP stack. It addresses problems such as anonymity, mobility, multi-homing, and end-to-end security.

The Host Identity Protocol architecture [25] defines a new global Internet namespace. Instead of an interconnected IP address and a domain name, a host is identified with Host Identity. It is not related to the host's position in the network topology like the IP address. As a host is identified with an unchanging identifier, a connection can be retained even when the host moves to a different branch in the network topology (host mobility). The transport layer uses Host Identities as endpoints while the network layer uses IP addresses purely as locators.

The Host Identity is a public key. It is represented by the Host Identity Tag which is a 128-bit long hash of the public key. A host may have a single or multiple host identifiers. Each tag is connected to information of the host's location which, in this case, is an IP address. The prevailing IP address of the host is used to route the packets. If the host has several interfaces, they can all be utilized at the same time (multihoming). Mobility and multihoming are discussed in the End-Host Mobility and Multihoming with the Host Identity Protocol RFC [29].

The Host Identity Protocol is composed of Diffie-Hellman key exchange, mobility exchange, and additional messages. With the HIP connection establishment, HIP hosts can get assurance that the peer they are communicating with has the private key corresponding to their host identifier, and so is who he claims to be. This base exchange creates a pair of IPsec Encapsulated Security Payload (ESP) Security Associations

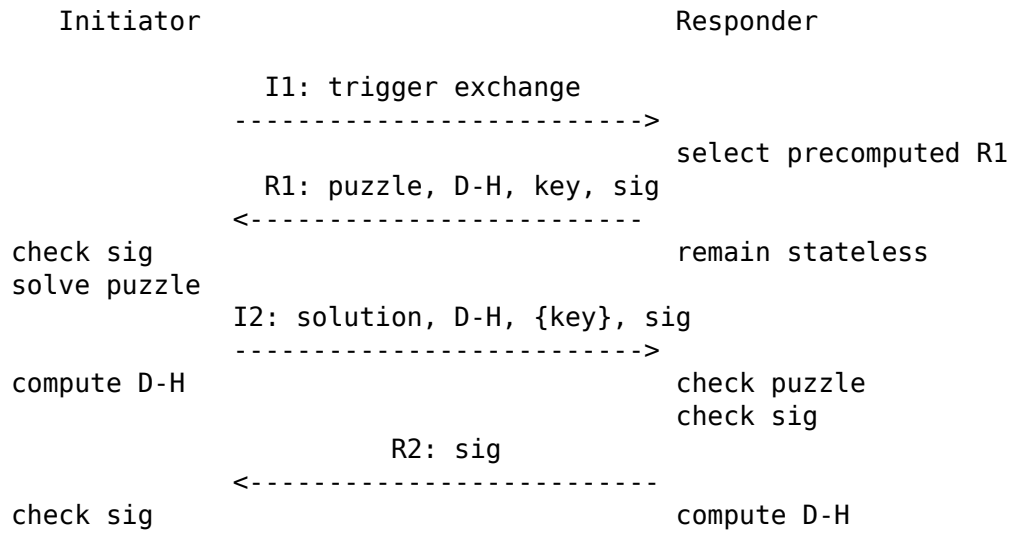

Figure 2.2: HIP connection establishment diagram from the Host Identity Protocol RFC [26]

(SAs), which can be used e.g. to encipher the connection later on. Using the ESP Transport Format with the Host Identity Protocol is discussed in RFC [14].

2.2.1 Connection establishment

The HIP connection establishment is described in the Host Identity Protocol RFC [26]. Figure 2.2 shows an example diagram of the handshake. The Initiator first sends a trigger packet, I1, to the Responder to begin HIP base exchange. When the Responder receives the I1 packet, it will remain stateless to protect itself from I1 replay attacks. The Responder replies by sending back the first response (R1) packet which contains a pregenerated puzzle which the Initiator must solve to generate the second initiative packet (I2). The Responder can easily adjust the difficulty of the puzzle even with it being pregenerated. The puzzle is a protection measure against replay attacks. Included in the message are also the initial Diffie-Hellman key exchange parameters, the Responder's host identity public key, and a signature, which covers part of the message.

When the Initiator receives the R1 packet, it checks the signature is from the Responder. It will then have to solve the received puzzle before it can send the second initiative packet (I2). The I2 message contains the solution to the puzzle, a Diffie-Hellman parameter required by the Responder, the HI public key, and the Initiator's signature covering the whole message. The Initiator can now compute the Diffie-Hellman key which is used to protect the connection from outsiders.

When the Responder receives the I2 packet, it checks that the solution is correct and

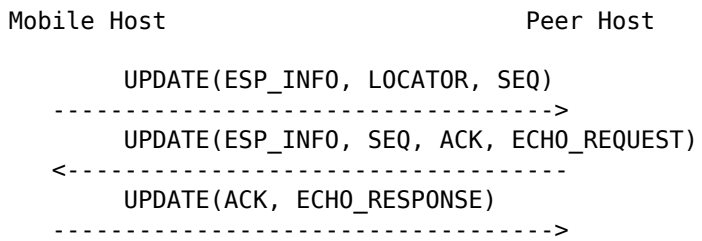

Figure 2.3: Readdress without Rekeying, but with Address Check [29]

that it is signed by the Initiator. If the information is correct, it will change its state to "HIP association established" and send the final signed R2 packet that concludes the base exchange for its part. The packet only contains the signature calculated over the whole HIP envelope. After this the Responder will compute the Diffie-Hellman key now shared by the two hosts. When the Initiator receives the R2 packet, it will check the packet's signature and change its state to "HIP association established".

2.2.2 HIP Mobility

By separating the transport layer from the internetworking layer, HIP enables network mobility and host multihoming solutions. HIP messages may contain a LOCATOR parameter to notify peers on alternate addresses for the host.

A simple scenario of handling a host's IP address change is presented in the End-Host Mobility and Multihoming with the Host Identity Protocol RFC [29]. Figure 2.3 shows an example diagram of the situation.

In this situation, the mobile host gets disconnected from the original network. It then acquires a new IP address from the current network and sends an UPDATE message to the peer host containing the new IP address in the LOCATOR parameter. The LOCATOR parameter also includes a lifetime for the locator. The UPDATE message also includes an ESP_INFO parameter which contains the old and the new SPIs (Security Parameter Indexes) for a security association. In this case they will both be the pre-existing incoming SPI. In addition the message contains the SEQ parameter indicating the peer host has to acknowledge the UPDATE. The mobile host waits for acknowledgement for the UPDATE from the peer host and will send the message again if no response is received in a while.

2.2.3 HIP Multihoming

A basic scenario of host multihoming is presented in the End-Host Mobility and Multihoming with the Host Identity Protocol RFC [29]. Multihoming means a host has several network interfaces connected to different networks. Figure 2.4 shows an ex-

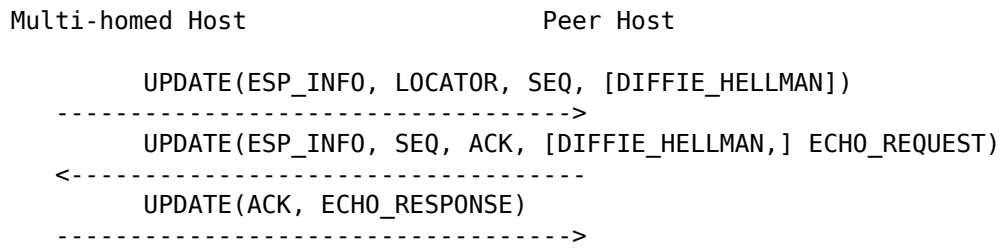

Figure 2.4: A basic multihoming scenario [29]

ample diagram of the situation.

The multi-homed host will send an UPDATE message with a LOCATOR parameter and the ESP_INFO parameter. It requests a new security association for the new address by setting the OLD_SPI field of the ESP_INFO parameter to zero and the NEW_SPI to a new value. In the LOCATOR parameter, it will include both the new locator and the new SPI value, and the original address and the SPI associated with it. It will also set the Preferred locator field in the new locator's sub-parameter to one, and that of the original locator to zero. It may also send new Diffie-Hellman parameters in the UPDATE message.

The peer host receives the UPDATE message and validates it. Then it creates a new security association for the new address and sends it in the ESP_INFO parameter of a response UPDATE message to the multi-homed host. After receiving the message, the multi-homed host will send another UPDATE message to the peer to verify the second address.

Message Sequence Chart, commonly used in illustrating protocols, was used in this chapter to illustrate the message exchange. In the next chapter I will present some recent developments in the field of software visualization and especially network and protocol visualization.

Chapter 3

Software visualization

Computer science educators generally agree that "Using visualizations can help learners learn computing concepts" [27]. Research indicates that promoting learner engagement is more important than graphics by itself [10, 11].

A working group at ITiCSE 2002 identified six levels of learner engagement in software visualization: no viewing, viewing, responding, changing, constructing, and presenting [27]. No viewing means learning without the aid of any visualizations. Viewing is part of all the other forms of engagement. By itself it is the most passive form of engagement. Responding is answering to questions related to the visualization. An example of changing is allowing the learner to modify the input of an algorithm under study. In Constructing learners map e.g. an algorithm to visualization by simulating an algorithm. Presenting entails presenting a visualization to an audience for discussion.

In addition to listing levels of engagement [27] listed common reasons for not using visualizations in education: it takes too much time to create visualizations, it takes too much time to find and learn the tools, visualizations take too much time from the lecture, and last, teachers are concerned about the equipment needed in visualizations. The previous are important considerations to be taken in account when creating a new visualization: It must be easy and save time. Next, I present software visualization and network visualization tools.

3.1 Tools and guidelines

In this section I will present various attempts in software and algorithm visualization with the focus being in teaching them.

Shaffer et al offer an implementors guide for creating visualizations based on their study of over 350 visualizations [36]. They stress a well-defined goal, a simple and consistent user interface, animated complex operations, data sets, and ease of running.

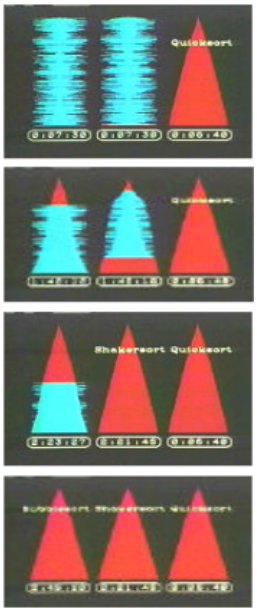

Figure 3.1: The race of three Exchange Sorts in Sorting Out Sorting video. [2] [3]

Some things can be demonstrated quite well by means of animation and a narrative, as Ron Baecker's Sorting Out Sorting film [2] and its success [3] demonstrate. As is noted in several studies, recommended by Baecker in [3], an algorithm animation should be abstract and highlight the essential parts of an algorithm. Showing too many details or program code, which is the focus of program visualization in e.g. [34], can cloud the underlying algorithm in teaching basics. Comparing the performance of different algorithms as in Figure 3.1 is often exciting.

Furcy, Naps and Wentworth suggest in [8] that being able to simulate an algorithm, without the need to actually program can further strengthen learning. They transform Baecker's original video into a highly interactive version for the web.

Brown and Sedgewick presented the Brown University Algorithm Simulator and Animator (BALSA) in 1985 [4]. It was designed to be used mainly with an instructor conducting or controlling use, which is one of the intended use cases for my visualization too. They made some useful observations while the system was used on several courses: When presenting new concepts or algorithms, it is important to start with a small case first, otherwise there is too much confusing clutter on screen. Graphical cues for state information were found useful on atomic pieces of data structures. When used consistently, they could be used to tie elementary and complex situations together. Elementary cases have more room for explanatory text and the cues become familiar while complex cases do not have room for so much text but the familiar cues are there. Multiple views support was useful for comparing different algorithms for the same data set. Multiple views in BALSA were also found useful for studying a single case using different graphical representations, which could present the situation

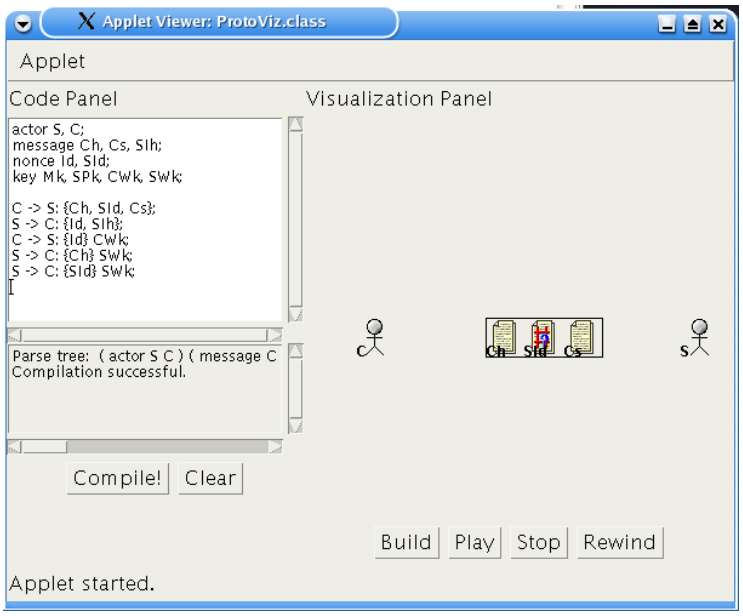

Figure 3.2: Screenshot from ProtoViz [7].

on different levels of abstraction or highlight different aspects of the data structure. Scripting turned out to be a very important part of the BALSA system. It made it possible to create presentations beforehand mixing different representations. Creating presentations needs to be as easy as possible.

Schweitzer et al developed a tool called GRASP to aid in teaching security protocols [35]. GRASP concentrates on illustrating protocols and attacks. For example, Figure 3.3 presents Diffie-Hellman key exchange protocol with an eavesdropper listening. They devised a protocol specification language, like the one in ProtoViz [7], to describe situations. Figure 3.2 shows a screenshot from ProtoViz. Both GRASP and ProtoViz are used by writing scripts. GRASP also supports step-animation. Scripting provides generality but requires that the user learns and follows the syntax. What I wanted to do is provide buttons for actions, so no coding or scripting is generally required. Of course this limits use to supported protocols and situations.

3.2 Network visualization

Understanding how a single machine works can be hard, but it can be even more hard if a group of machines work together over a network. Analogies can help in understanding difficult concepts.

Teaching by analogy means mapping a new concept to a familiar concept and by so doing, making it easier to understand the new concept. This is taken further in [24]:

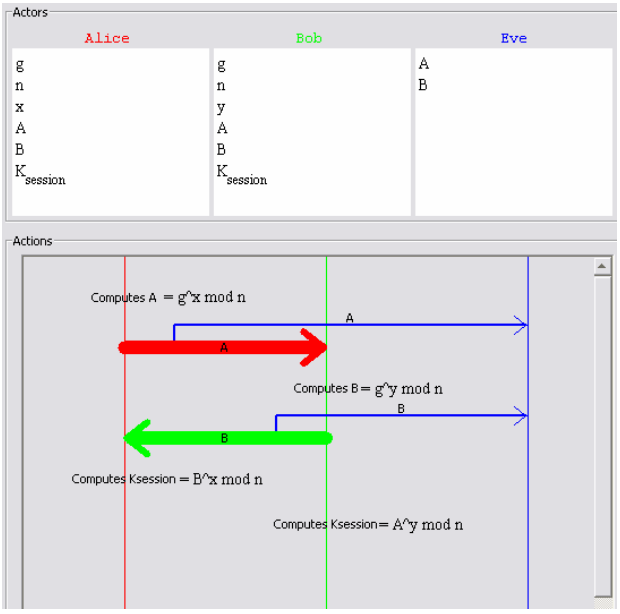

Figure 3.3: Diffie-Hellman example in GRASP [35].

People are receptive to stories as they have been a means of transferring information among people for thousands of years. The more extravagant the story, the easier it clings to our minds. The method presented in the paper starts by identifying a difficult concept, then finding an analogy familiar to the target group, and finally creating an exaggerated story around the analogy. As an example a story is presented using the Pony Express as an analogy to TCP/IP and another of a chief's sons creating rules for communicating around the campfire analogous to IEEE LAN protocols. Visualizations can be created to support such stories but the method relies heavily on the lecturer's presentation. The visualizations are also connected to the story in case and not re-usable.

Visualization in itself is creating analogies. Using a common visual catalog supports learning since once a student has learned the analogies in one case, he can more easily understand other cases where the same analogies have been used. If one were to create a new visualization with new analogies, it would benefit if they were applicable to other cases as well. This is one of the reasons for using the general visual analogies that are used in my system.

My figure 3.4 is a visualization exercise of Diffie-Hellman key exchange using puzzle pieces as an analogy. Diffie-Hellman key exchange is used, among other things, in the Internet Key Exchange protocol (IKE) of the Internet Protocol Security protocol suite (IPsec). It is also used in HIP, which acts as an example in this thesis. In the previous protocols, Diffie-Hellman key exchange is used to create a shared secret between two peers over an insecure communication channel. This shared secret can then be used for creating a symmetric session key more suitable for data transfer. The figure repre-

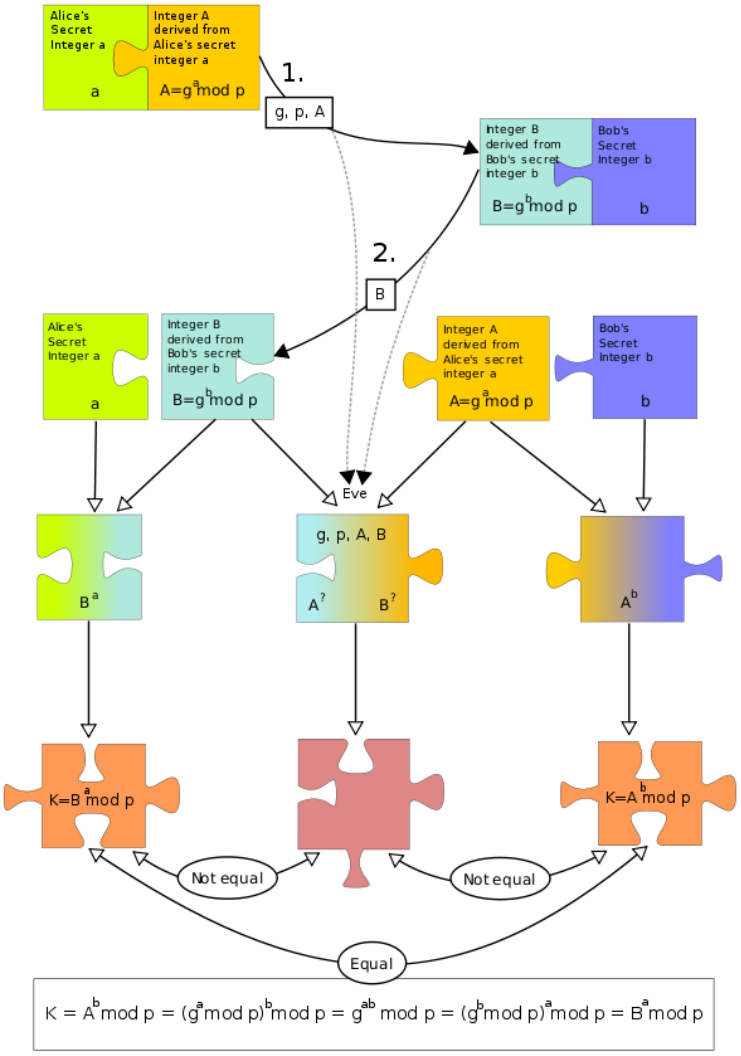

Figure 3.4: Diffie-Hellman key exchange analogy.

sents the exchange of values needed in Diffie-Hellman to create the shared secret, and the attempt of a third party to create the same key by eavesdropping on the message exchange. The puzzle pieces illustrate Diffie-Hellman values. The coinciding tabs and blanks of the pieces illustrate how values are derived from other values. The forming of new pieces represent mathematical functions used to create the shared secret. Using this sort of visual analogy would need to be used broadly and consistently in several protocols to be useful, since the analogy is not self-evident.

Student-active methods are praised in many papers referred to in [24] and many of these have roots in the Socratic Method in which a group of students are posed questions and by so doing encouraging student discovery [5].

“– this technique allows students to present the topic with their level of understanding

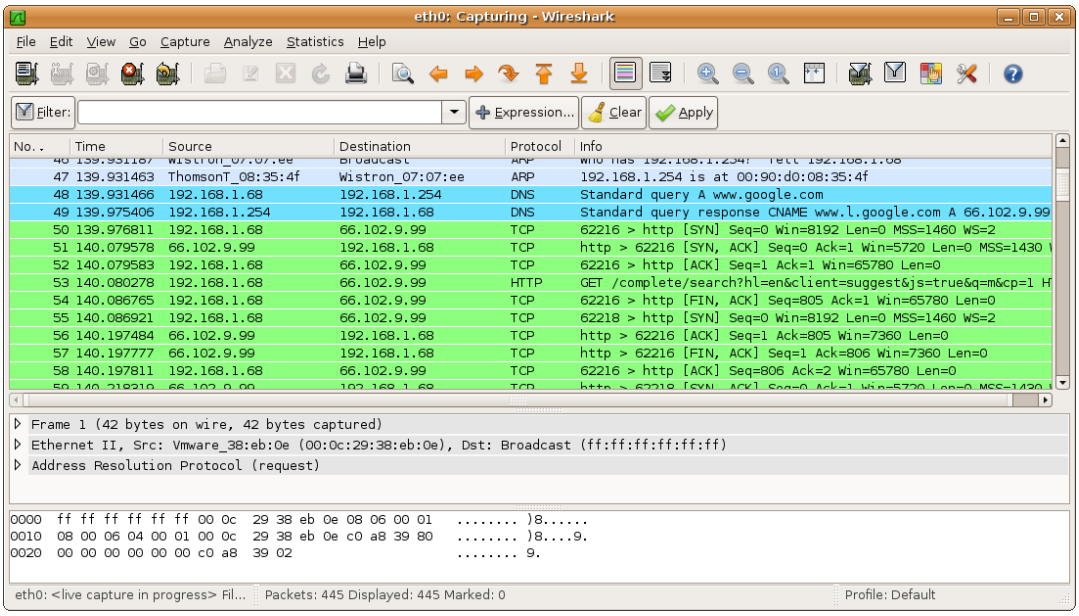

Figure 3.5: Screenshot from Wireshark [6].

to others on similar levels. In today's collaborative learning exercises, students are typically posed a question, asked to formulate an opinion, and then expected to convince other students of their decision. This method allows the students to test the strength of their belief by forcing them to defend it in the face of criticism." [24]

Message exchange of protocols is often demonstrated via Message Sequence Charts (MSC) [12], like I have done in Chapter 2. As protocols often send packets through a network or networks, packets take their time to travel if they even get through. Message Sequence Charts are not very good at demonstrating this, as they represent instantly travelling messages.

Network traffic analyzing software like Wireshark [6] can be used successfully in teaching [9] but they are more suited for advanced examples. Wireshark displays so much information it can be difficult to make sense of it before one is familiar with the protocols displayed and the software. Figure 3.5 is a screenshot of Wireshark and Figure 3.6 is a screenshot of HIP base exchange from an older version of the software.

Oechsle et al developed the VisuSniff [30] software which could be used to visualize network traffic with teaching in mind. It could load TCPDUMP data or capture data on-the-fly. Protocols and/or hosts could be filtered by clicking them in the overview diagram (Figure 3.7). The sequence diagram could be browsed (Figure 3.8) packet by packet and individual packets opened for further examination (Figure 3.9). Handling actual network traffic in teaching has its advantages and it has been done quite well in VisuSniff. It limits the cases to actual live or recorded situations.

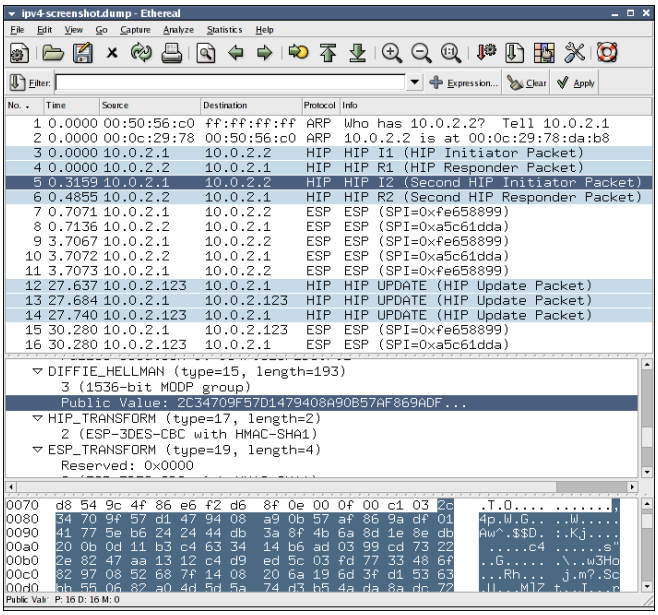

Figure 3.6: HIP on Ethereal. [6]

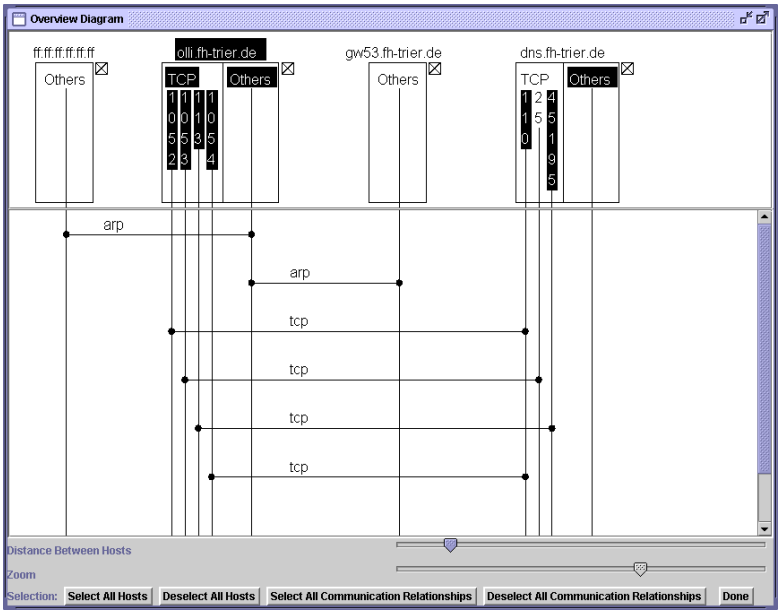

Figure 3.7: Overview diagram from VisuSniff [30]: hosts, ports, and their communication relationships (selected items inverse).

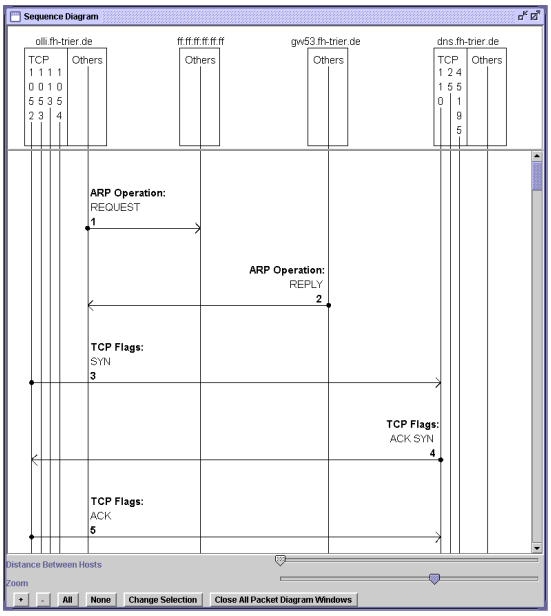

Figure 3.8: Sequence diagram from VisuSniff [30]

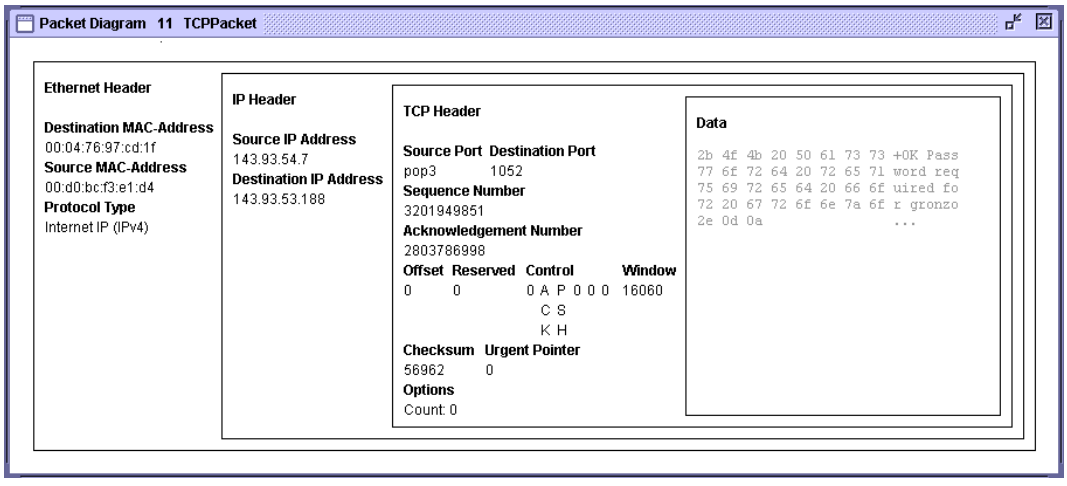

Figure 3.9: Packet diagram showing a TCP segment in VisuSniff [30]

Chapter 4

The Matrix Visualization Framework

Matrix is a framework for interactive software visualization which enables algorithm animation and simulation [22]. Visualization makes it possible to hide implementation-level details making it easier to understand the underlying concepts. Section 1.2.2 presented a scenario in which a student uses the system for studying by herself. Internally Matrix handles the actual data structures, and this makes automatically assessing user generated simulations possible. The system has all the necessary tools for easily creating exercises for students, assessing them and giving points for the exercises built in. Matrix can randomize the input data for varying automatically generated exercises. By doing these exercises, the student can check his understanding of the subject, and, as feedback is instant, he can make sure he is not left with any misunderstandings [20]. Automatic assessment can also save resources on larger courses as less work is required for creating and assessing the exercises. As the system can be used online in a browser, the students do not have to go through a strenuous installation process. In addition, they are not bound to any specific place or time for using the system.

Time is modelled and represented in Matrix as steps. This allows triggering events with precision. The user can move back and forth in time through the steps. Each step may display information or questions related to just that situation.

Similarly to BALSA, Matrix also supports multiple views and different simultaneous graphical representations. I also added graphical cues and support for different representations of elements to protocol simulation in Matrix.

4.1 TRAKLA2

TRAKLA2 [21] is a web-based learning environment for automatically assessed visual algorithm simulation exercises that is built on Matrix. The number of data structures and algorithms included in Matrix can be visualized in TRAKLA2. TRAKLA2 can be used for individually tailored exercises which are automatically evaluated. In the exer-

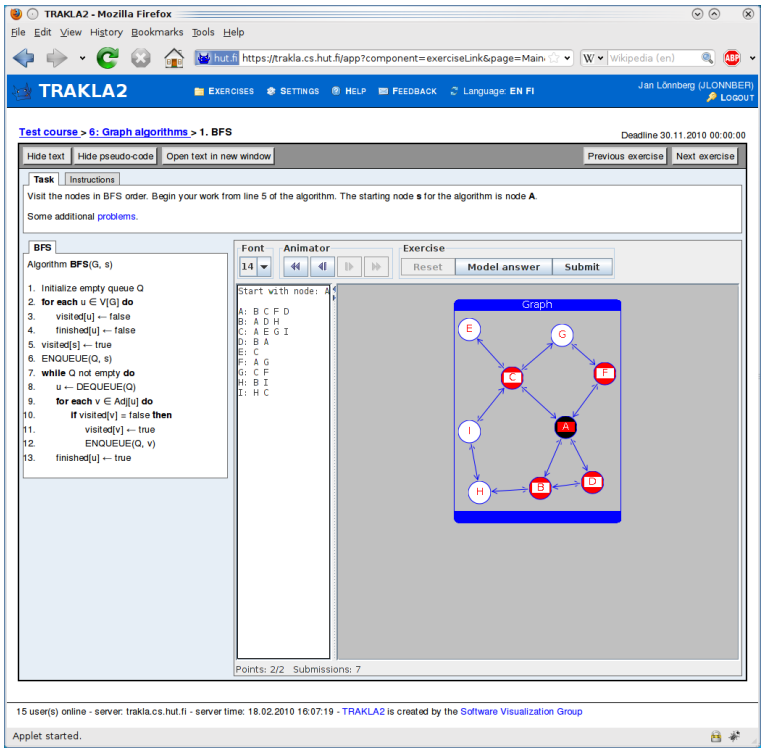

Figure 4.1: A typical use scenario in TRAKLA2.

cises, the user simulates what an algorithm would do by directly manipulating visual representations of the data structures that the algorithm works on. The interactions can be triggered e.g. by clicking or dragging and dropping.

Figure 4.1 is a screenshot of a typical use scenario in TRAKLA2. The user is completing a breadth-first search (BFS) exercise. The task description is shown at the top, a description of the BFS algorithm on the left, and a visualization of the current situation on the right. Visible are also the controls for moving back and forth in time, and buttons for resetting the exercise, displaying a model answer, or submitting the exercise. The user may ask for grading of the answer, which will check the user's answer and give immediate feedback, such as how many steps of the algorithm were right. After this the user may be asked if he wants to submit the answer or try again with new values. Checking of the answer may be restricted to e.g. one grading before submission of the answer. Resetting the exercise will initialize it with new start values, using a pseudo-random seed. The user may also view the model answer to the exercise, where he can go back and forth through the steps. After viewing the model answer, he will need to reset the exercise, so he can test his understanding, and be assessed.

Matrix provides a general-purpose platform for visualizing data structures and algorithms. It is designed to be easily extended, so that adding new data structures, algorithms, and visualization models is possible.

On the implementation level, Matrix handles concept interfaces and probes which implement those interfaces. New data structures can be added to Matrix by implementing an interface. There is a corresponding interface for each visual concept, such as a graph or a tree. To enable animation, data structures need to store instance variables in Matrix memory structures. This way previous situations can be restored.

Matrix provides the following fundamental data types (FDT): arrays, linked lists, common trees, binary trees, and graphs. They are used in implementations for many basic data types, search trees, priority queues, and so on. These are called conceptual data types (CDT). All fundamental data types have visual counterparts that enable the visualization and animation of the structure. As all conceptual data types are derived from the fundamental data types, they too can easily be visualized. Simple and complex operations may be used in the data type implementations.

Creating new exercises in TRAKLA2 involves a straight-forward work flow defined in [21]. The first thing needed is a manuscript of the exercise. A programmer will then create the exercise based on the manuscript by defining a Java class for the exercise. The following things need to be defined:

- data structure types employed in the exercise
- names for the visual representations
- layout for each visual representation
- layout for representations in the user interface
- methods to create randomized initial values for the data structures
- an algorithm to create a model solution for a specified input
- push buttons for the exercise
- allowed/disallowed user interface operations for the exercise

4.2 MatrixPro

MatrixPro [16, 17] is a tool for on-the-fly demonstration of data structures and algorithms. With it, a user may create presentations e.g. for a lecture without the need to code anything. The presentations may be prepared in advance or on the fly using visual controls. Ready-made operations can be used to simulate the working of real algorithms. The lecturer can easily respond to “What if” questions posed by students in a lecture. Section 1.2.1 presented a scenario where a teacher uses MatrixPro to visualize Host Identity Protocol.

Chapter 5

Architecture and Design

By extending the Matrix platform we can use it for protocol simulation as well, and take advantage of the features it offers and software, such as TRAKLA2 and MatrixPro, which run on Matrix. Using TRAKLA2, it is possible to create exercises for students for protocol simulation where the system asks the student, for example, questions such as "What happens in the case below?", "What will the sequence number of the packet be?", "What will the host do now?", " Will the host send an ack?", "Where will the host get its IP address?", "To what state will the Responder move now?", or "Which information elements will the Initiator send as reply?".

Lönnberg's and Koivikko's pilot version [18] for Matrix experimented with visualizing ARQ (Automatic Repeat reQuest) protocols and their use by TCP. The pilot also introduced an interactive version of the Message Sequence Chart (MSC) as a visualization scheme in Matrix. MSC is often used for visualizing protocols e.g. in the print media. The pilot proved that visualizing protocols was possible using Matrix. I first started transforming the pilot's visualization to meet the requirements stated in this thesis. The idea of visualizing the hosts as boxes, with state information and control buttons, seemed to work in the pilot, so I used it in my own derived visualization. A modified version of the MSC seemed like a good idea as well. A screenshot from the pilot shown in Figure 5.1 has two hosts and arrows depicting messages like in a MSC. I tried to take the best ideas from the literature, and modify the pilot's visualization based on them. Later, I decided it was best create new interfaces instead of using the ones used in the pilot.

5.1 Architecture

In my visualization, data communications protocols are viewed on a message mediation level. The user can send messages and control what happens to the packets simulating different situations and circumstances. The interfaces are divided into two

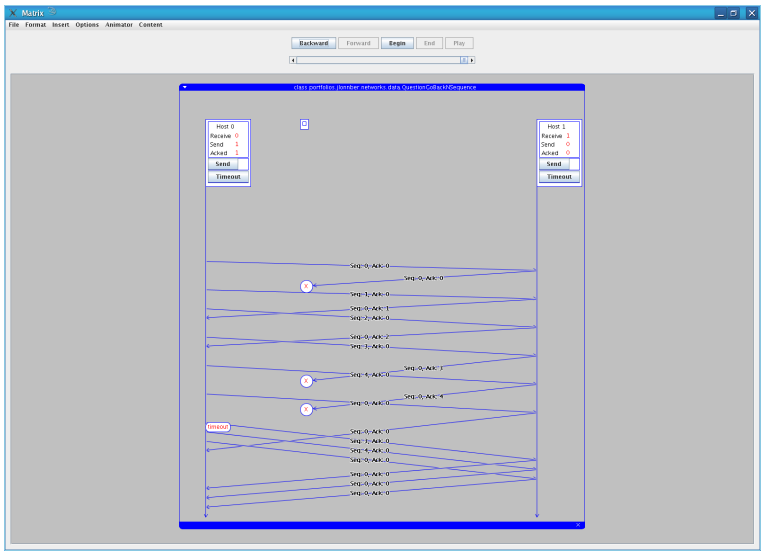

Figure 5.1: Pilot version of protocol visualization for Matrix [18]

groups: data and view. On the data side, the network sequence interface extends the fundamental data type interface of Matrix. It represents a general network sequence involving an arbitrary number of hosts and events. The HIP sequence interface extends this with HIP specific hosts and events. Figure 5.2 shows the interfaces for the data side.

The visualization of protocols can be defined as individual cases illustrating different aspects of the protocols, such as connection establishment, data transfer, mobility, and multihoming. When the case to be studied is known, this knowledge can be taken advantage of in the visualization by showing elements important to the case in question and hiding non-relevant elements. If the goal of the visualization is known beforehand, also elements a host strives to know may be visualized. Next, I will discuss the different architectural components of the design. The general components are discussed first, then example components for Host Identity Protocol are discussed.

5.1.1 Network sequence

This interface represents a basic network sequence where there are hosts and message exchange between them. It extends the fundamental data type (FDT) interface of the Matrix platform. It consists of hosts involved in the network sequence and an array of the events.

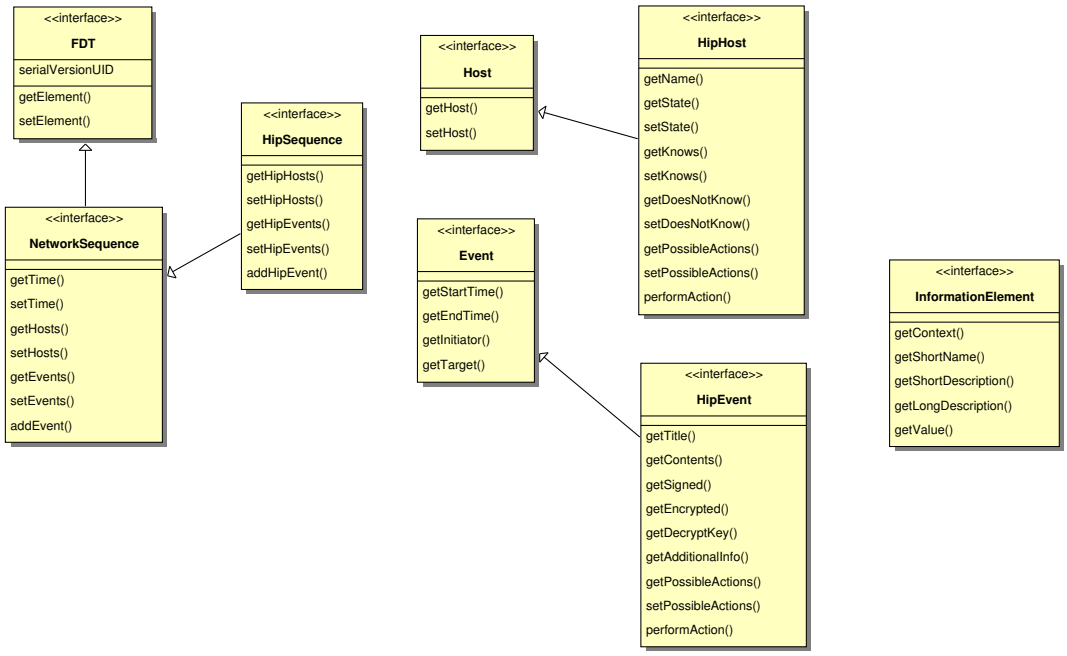

Figure 5.2: An UML class diagram of the data interfaces for the visualization

HIP sequence

The HIP sequence interface represents a basic HIP sequence. It extends the Network-Sequence interface, and consists of HIP hosts and an array of HIP events.

5.1.2 Host

The Host interface represents a general network node and, as such, does not involve specific information, but it can be extended for the needs of different protocols. The reason it is so reduced is because protocols are so varied, that hosts associated with them may have very little in common.

HIP host

The Hip host interface which extends the Host interface defines a name, a state, a list of known pieces of information, a list of unknown pieces of information, and a set of actions that are applicable in a given situation. The suggested default names for the hosts are Alice, Bob, Eve, and Jack, the archetype characters used in cryptography. Every object has a role in the network running the protocol. For example, hosts Eve and Jack relate to an eavesdropping person, or a person trying to hijack the connection. Eve will automatically receive everything sent, even though she is not marked as the

receiver, as she is eavesdropping on the connections. This is because, with the network technology in use today, the physical networks are mostly broadcast networks in which all the messages really are for anyone to see and hear. Also Jack will automatically receive all the messages sent and he will try to fool the peers into thinking he is someone else. The possible actions for a host are "Send" and "Timeout". The possible states for a HIP host follow that of a HIP state machine. They are

- 0 = UNASSOCIATED: State machine start
- 1 = I1-SENT: Initiating base exchange
- 2 = I2-SENT: Waiting to complete base exchange
- 3 = R2-SENT: Waiting to complete base exchange
- 4 = ESTABLISHED: HIP association established
- 5 = CLOSING: HIP association closing, no data can be sent
- 6 = CLOSED: HIP association closed, no data can be sent
- 7 = E-FAILED: HIP exchange failed

5.1.3 The known and unknown pieces of information

The lists of known and unknown pieces of information elements can be useful in teaching protocols. They visualize the intent of a host. In HIP, the lists consist of, e.g., host identity tags, and Diffie-Hellman key exchange values. The HIP hosts also keep track of the state of the addresses of the peer hosts. Possible values for the states are VERIFIED, UNVERIFIED, and DEPRECATED. For other protocols the lists consist of information elements specific to those protocols. The items in the lists are defined in the Information element interface. By utilizing the knowledge of the situation being visualized and comparing the lists to a list of items that can be known in that situation, the lists can be automatically updated according to received pieces of information.

5.1.4 Event

The Event interface represents a general network event which takes place between two parties, the initiator and the target. The event embodies the life cycle of a packet from sending to arrival or disappearance. The interface also defines the start and the possible end time of an event. The interface is meant to be extended according to the needs of the protocol it is to be used for. See HipEvent for an example. Last, the interface defines the possible outcomes to the event which are that the packet arrives to the target or is lost.

HIP Event

The HIP event interface extends the Event interface by defining a title for an event, such as “Initiative message 1” or “I1”. It defines a list of pieces of information the message contains. It also defines what part of the content is signed and what is encrypted, and which key is needed to decrypt the encrypted content. The interface also defines additional information concerning the event, and for different phases of the event: when the message is sent, and when it arrives to its destination. This information may be displayed below the hosts and between the hosts.

5.1.5 Information element

The information element interface represents a distinguishable piece of information. It is meant to act as a common element for all protocols. It defines an actual value for the information element in question, but this is not always needed in the common use cases of the interface. Often useful properties of the information element are: the short name, the short description, the long description, and the context. These can be utilized in different situations when visualizing the information.

Context Pieces of information related to each other can be grouped and visualized differently by utilizing the defined context. The default context groups are general, security related and Diffie-Hellman related.

Short name The short name can be for example an abbreviation. For example for “Alice’s Host Identity Tag” the short name could be “HIT A”. Showing just the short name can simplify the view in a situation where the information elements are familiar to the user. It can be used side-by-side with the short description when the element is not previously known. It can also be useful when the visualization scheme has little room for representing the element.

Short description The short description for an information element should be shown when the element is not familiar to the user previously. It can be shown beside the short name in the list of known information elements of the host. An example value for this would be “Bob’s Host Identity Tag”.

Long description The longer description for an information element contains a more thorough explanation of the information element. It can be displayed for example in a pop-up window when the information element is clicked. The content of this field can contain several paragraphs and may contain simple links to URLs. An example value for this in HIP would be:

> "The Host Identity Tag is a 128-bit value – a hashed encoding of the Host Identifier. There are two advantages of using a hashed encoding over the actual Host Identity public key in protocols. Firstly, its fixed length makes for easier protocol coding and also better manages the packet size cost of this technology. Secondly, it presents a consistent format to the protocol whatever underlying identity technology is used.
>
> The Host Identity Tag is a type of ORCHID, based on a SHA-1 hash of the Host Identity, as defined in Section 2 of RFC4843." [RFC5201]

5.1.6 General views and colors

To help understand what is happening and how the protocols work, the following means are used. The state of the involved parties is visible at all times. Messages are color coded and clear arrows are used to illustrate which way a message is sent. Unavailable operations are not shown. The visual elements have been circled in figure 5.3.

5.2 Common visualizing guidelines for protocols

In this work, a host is visualized as a simple box, with a name for the host at the top, state information below it, and controls for sending packets and controlling timeouts at the bottom. With the state information available at all times, it is easier for the viewer to keep track of what is happening in the protocol message exchange and why. Since the action of sending a new packet or resending an old packet after a timeout are associated with a host, it is natural to place the controls for these actions in connection to the hosts. The elements described in this section have been circled in figure 5.3.

The visualization of a host embodies the essential state information for the relevant use case. The state information represented can be e.g. the state machine state, what the host knows, what the host does not know, what information the host tries to acquire, sent bytes, received bytes, and acknowledged bytes.

Controls for moving back and forth in time, effectively serving also as the undo and redo functions, are at the top of the visualization. The buttons, provided by the Matrix platform, are Begin, Backward, Forward, End, and Play. Below the control buttons, there is an interactive timeline. A vertical line attached to each visible host on the screen ending up to an arrow pointing down represents a time axis.

Packets, or Events, are visualized as slightly downward arrows beginning from the time axis of a host and ending up to the time axis of another host or, in case the packet is lost, to a big X between the hosts. Each arrow has a unique color to make it easier to distinguish it from the others. In addition, sent packets are given a running number

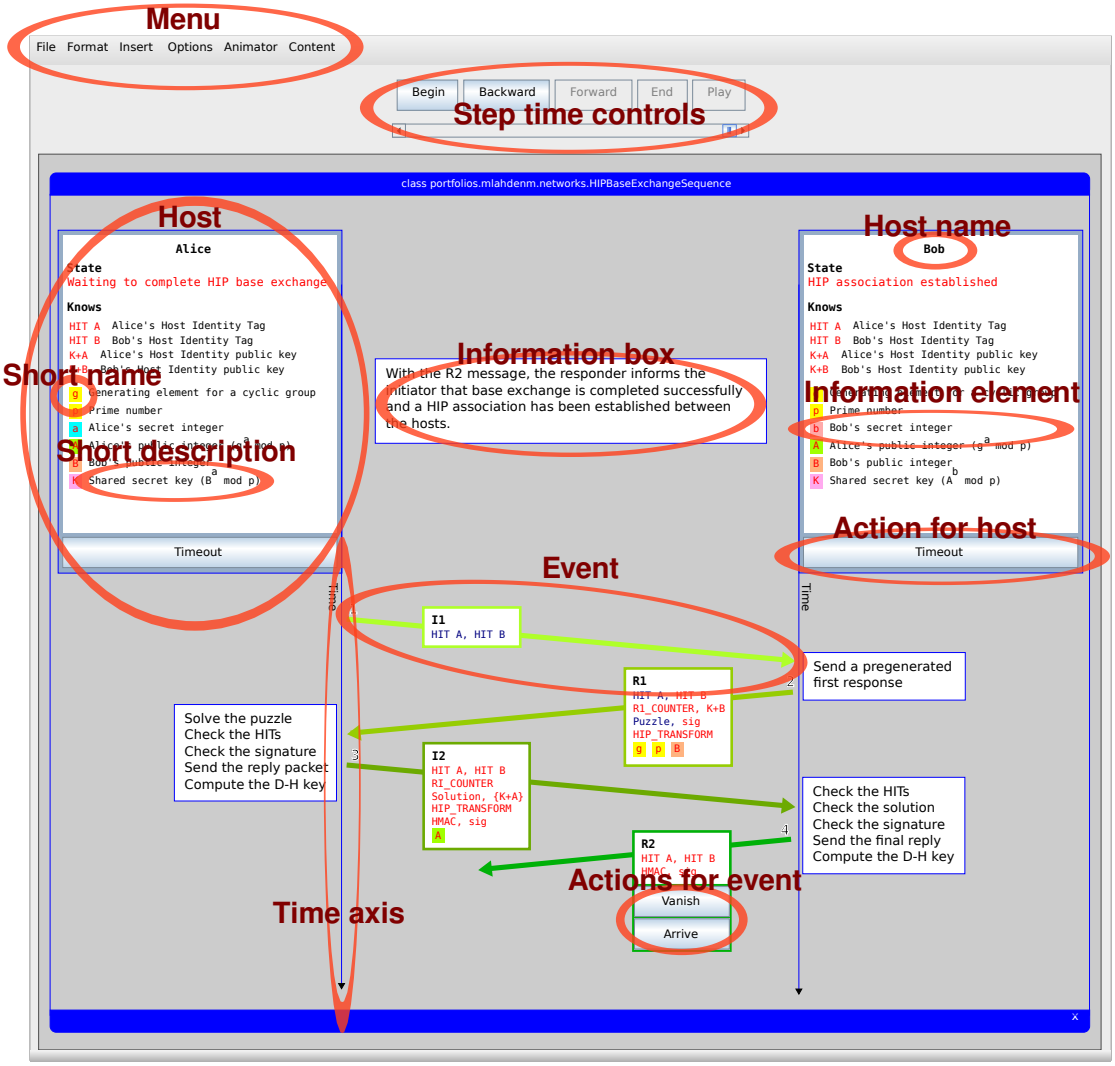

Figure 5.3: Different elements in my visualization

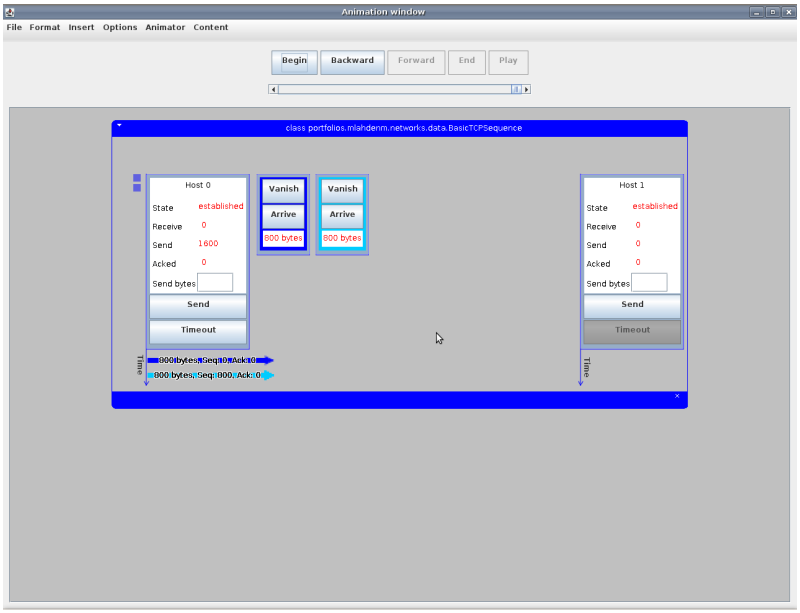

Figure 5.4: User interface sketch.

that is associated with the arrow. The packets' information is shown compactly in a box on the arrow near the senders end. This includes e.g. information sent to the other host, packet data payload size in bytes, the sequence number, the acknowledgement number, and the window size. A packet's message class affects the color theme of the arrow. Handshake related arrows follow a green theme and those related to closing a connection follow a red theme. The packet controls for simulating the network, that is controlling whether packets get to where they are going, are displayed at the bottom of a packet's information box. As a packet is received the new state of the receiver is updated, the knowledge that the receiver now has is updated in the box representing the host.

Data packets can also be visualized as simple colored boxes put in a column beside a host. They have a different color representing whether the corresponding packets have been sent, received, or acknowledged. It is also possible to visualize the send window and the receive window by framing a corresponding number of boxes to the window size. These sort of additional packet visualization boxes are visible in a user interface sketch in figure 5.4.

Beside the time axis, there may be information boxes stating what the hosts will do next. A larger information box can be visible between the hosts giving out context dependent information about the situation or information about elements the user has clicked. See section 5.4.1 below for reference.

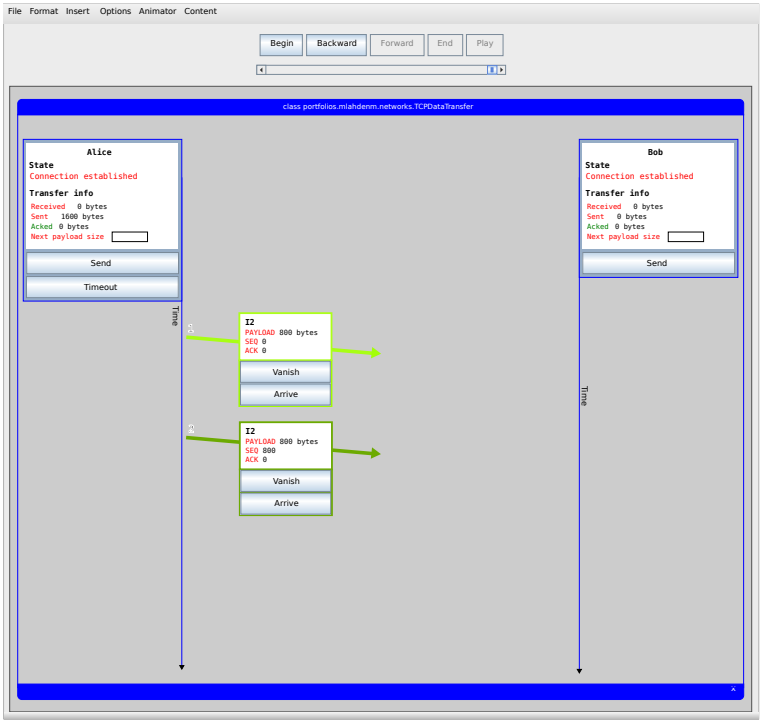

Figure 5.5: TCP data transfer

5.3 Transmission Control Protocol (TCP)

A description of the Transmission Control Protocol can be found in Chapter 2. Using the general visualization elements and design enables visualizing TCP just as well as HIP. Situations to visualize for TCP could be connection establishment, data transfer, and vulnerabilities. In data transfer one could visualize for example ordered data transfer, retransmission of lost packets, discarding duplicate packets, flow control, window scaling, and congestion control. Concerning retransmission, the different retransmission schemes and related protocols could be visualized. They are stop-and-wait, sliding window, go-back-n, and selective repeat.

TCP packets contain control bits also known as flags, and fields like the source and destination port, the sequence number, the acknowledgement number, window size, etc. The state of the TCP state machine can be displayed in the same way as with HIP. Connection establishment with TCP is very similar to HIP, though, more trivial. Figure 5.5 displays a basic TCP data transfer example using the visualization presented in this thesis. Here the values of the information elements are visible, and are shown instead of a short name used in many of the other figures.

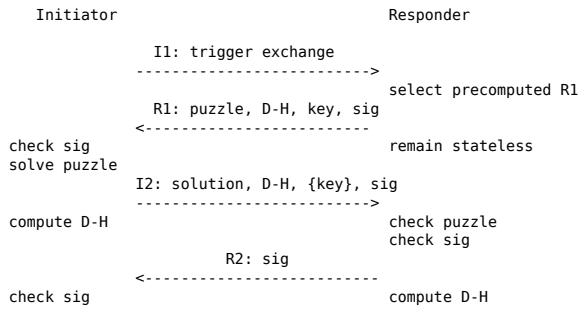

Figure 5.6: HIP connection establishment diagram from the Host Identity Protocol RFC [26]

5.4 Host Identity Protocol (HIP)

A description of Host Identity Protocol can be found in section 2. In this section I will walk through the HIP connection establishment, and then present mobility, multihoming, and man in the middle protection visualization examples. These visualizations can be used for example when giving a lecture about HIP, and they exhibit how the MatrixPro application, build on Matrix, can be used.

5.4.1 Connection establishment

Figure 5.6 presents a Message Sequence Chart about HIP base exchange from RFC5201 [26]. In this section, I will describe how the same is presented using my own visualization.

In the beginning Alice, the Initiator, wants to "Establish a connection" to Bob, the Responder. She knows her own Host Identity Tag (HIT) and that of Bob's. This situation is presented in Figure 5.7. Alice will next send a trigger packet (I1) to Bob. The packet only contains her own Host Identity Tag and that of Bob's. These are listed in the message arrow in Figure 5.8 presenting the situation.

After sending the packet Alice's state is changed from "Unassociated" to "Initiating base exchange" in the box on the left. A state diagram for the sender is presented in Figure 5.9. When the I1 message has been sent, an information box appears telling the following:

> The I1 packet begins the HIP base exchange. Opportunistic mode may be used if the Host Identity Tag of the responder is not known.

When Bob receives the I1 packet he will remain "Unassociated". In other words, he does not create a connection yet. This phase is presented in Figure 5.10. The information box has the following content:

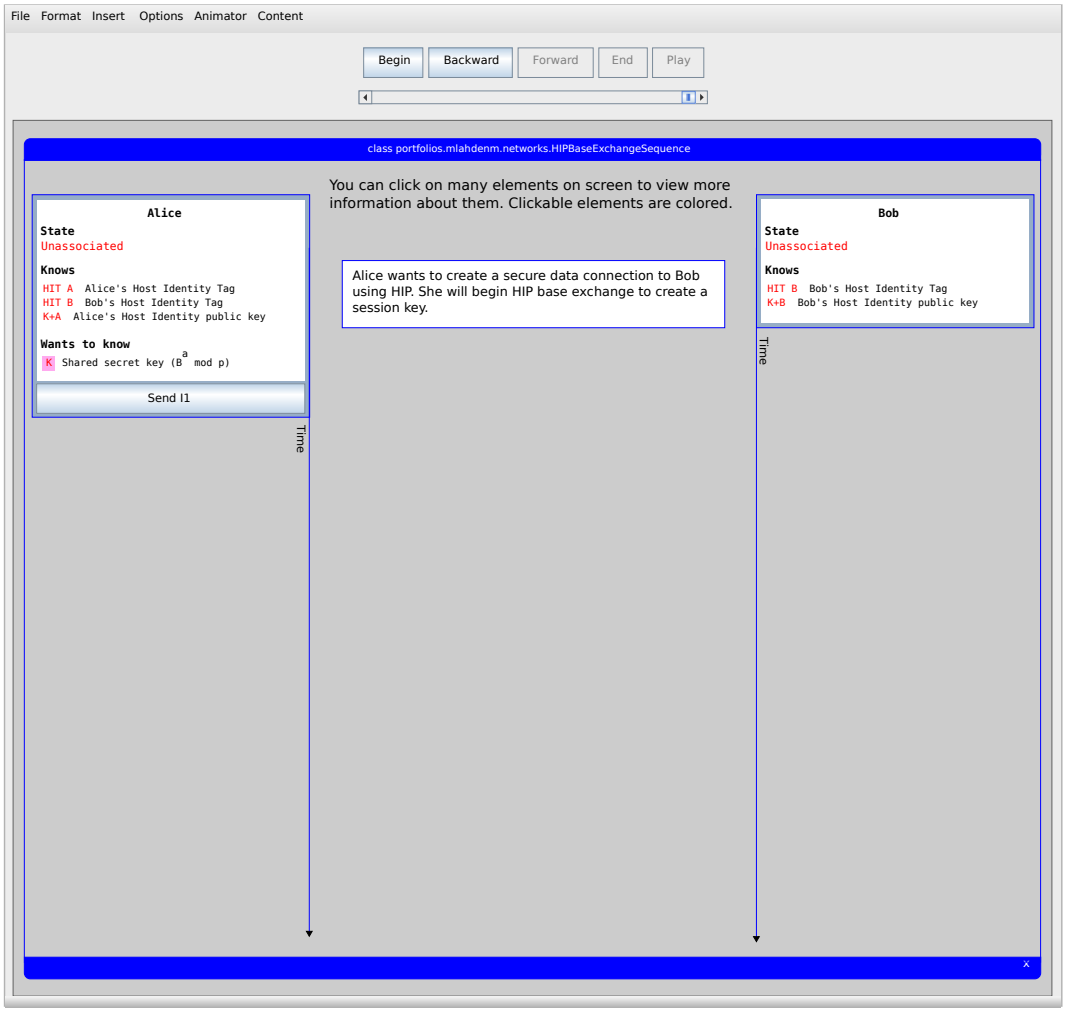

Figure 5.7: HIP base exchange.

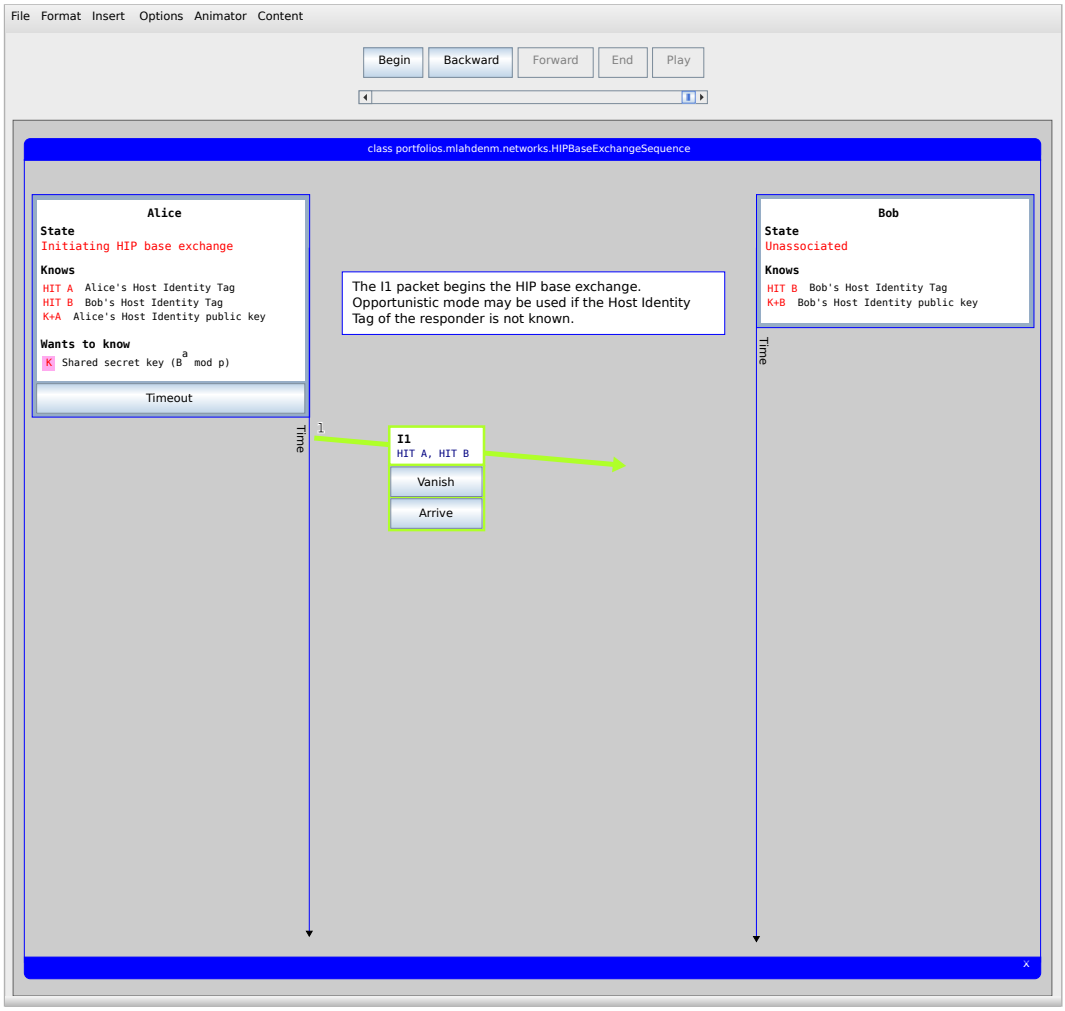

Figure 5.8: The first initiative packet (I1) begins the HIP base exchange.

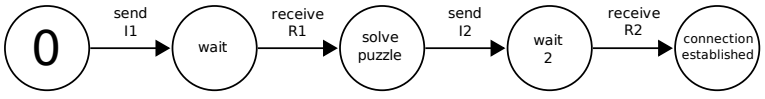

Figure 5.9: Simplified states of the sender in HIP base exchange.

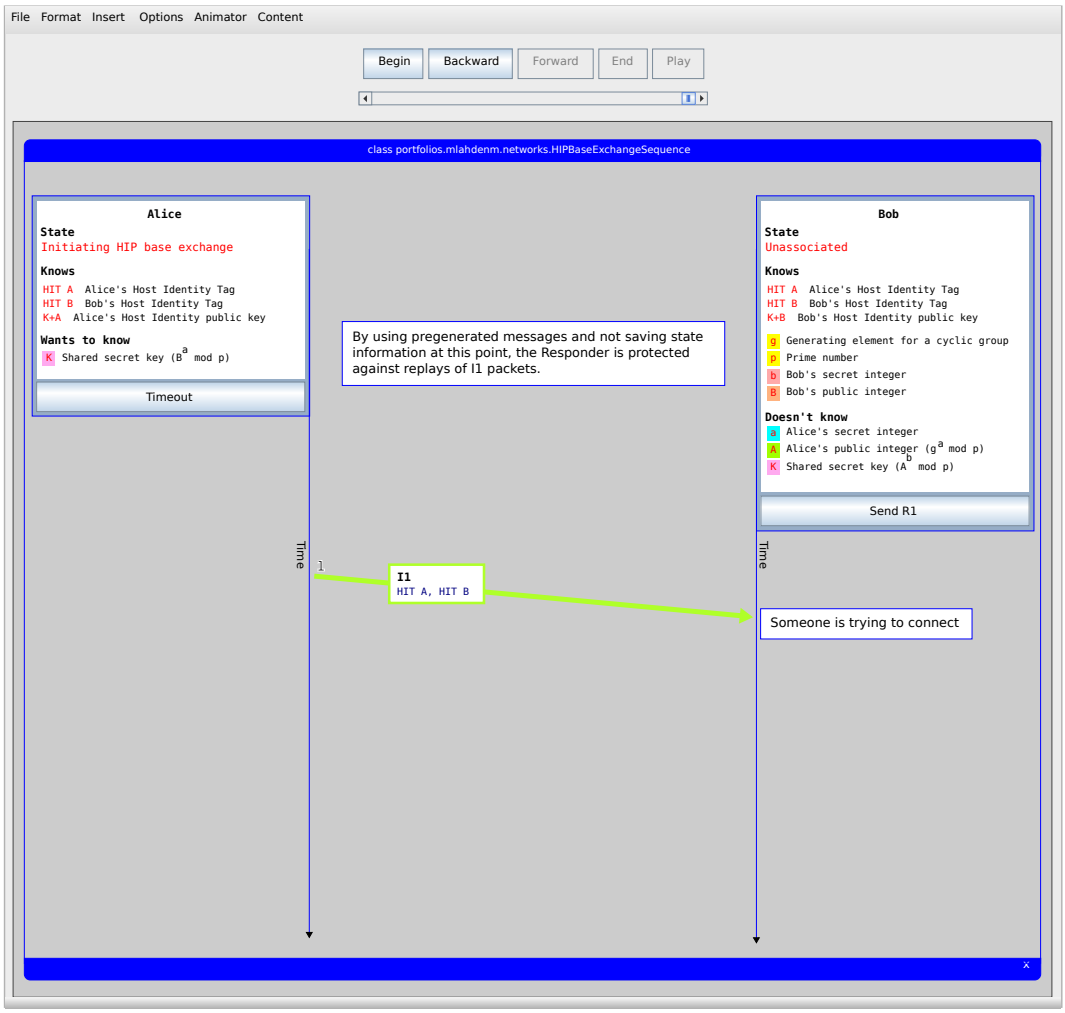

Figure 5.10: Receiving an I1 message, tells Bob someone is trying to establish a HIP connection with him.

> By using pregenerated messages and not saving state information at this point, the Responder is protected against replays of I1 packets.

Bob now knows “Someone is trying to connect” and will decide to “Send a pregenerated first response”. The first response (R1) packet, presented in Figure 5.11, contains a, usually pregenerated, puzzle which Alice must solve to generate the second initiative packet. Bob can easily adjust the difficulty of the puzzle. This is a protection measure against replay attacks. The message also contains an R1 generation counter for the previous puzzle. This is a protection measure against R1 replays for the initiator. Yet included in the message are the initial Diffie-Hellman key exchange parameters, Bob’s HI public key and a signature, utilizing the previous, which covers part of the message. Last, the message contains the encryption and integrity algorithms supported by Bob in the HIP_TRANSFORM field. All these are listed in the message arrow. After

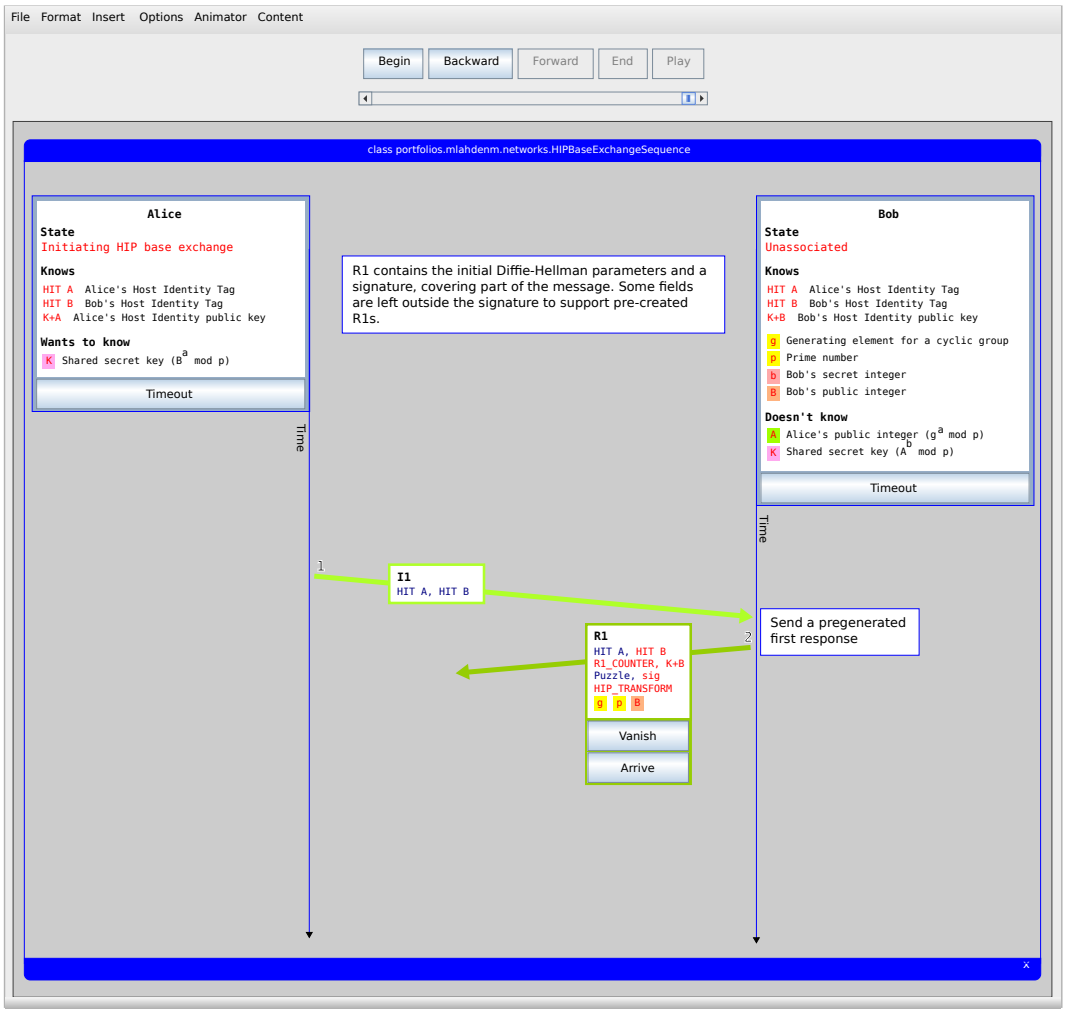

Figure 5.11: The first response message (R1) holds the first Diffie-Hellman parameters needed to create the session key at the end of the HIP base exchange.

the R1 packet is sent the state of the responding host remains “Unassociated”. The information box has the following description:

> R1 contains the initial Diffie-Hellman parameters and a signature, covering part of the message. Some fields are left outside the signature to support pre-created R1s.

When Alice receives the R1 packet, she will have to solve the received puzzle and then send the second initiative packet (I2) to Bob. This is presented in Figure 5.12. The I2 message contains the solution to the puzzle, a Diffie-Hellman parameter required by the Responder, the HI public key and a her signature covering the whole message. Alice’s state is now changed to “Waiting to complete HIP base exchange” and she will

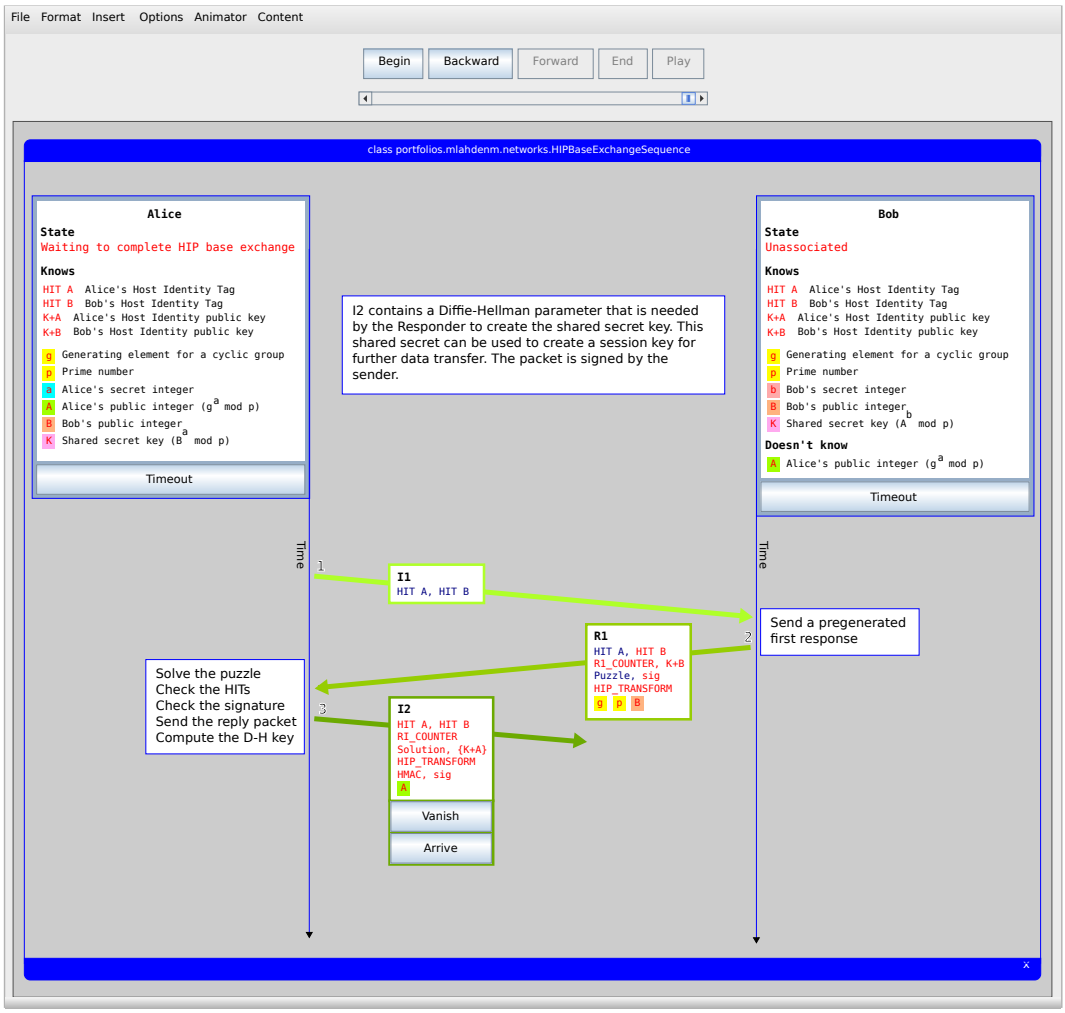

Figure 5.12: The I2 message contains Alice's public integer, and after it gets to Bob, both the hosts will share a common secret.

now compute the Diffie-Hellman key. As the message is sent the following information is displayed:

> I2 contains a Diffie-Hellman parameter that is needed by the Responder to create the shared secret key. This shared secret can be used to create a session key for further data transfer. The packet is signed by the sender.

When Bob receives the I2 packet, visible in Figure 5.13, he checks that the solution is correct and that it is signed by Alice. If the information is correct he will change his state to "HIP association established" and send the final signed R2 packet that concludes the base exchange for his part. This is presented in Figure 5.13. The packet only contains the signature. A state diagram for the receiver is presented in Figure 5.14.

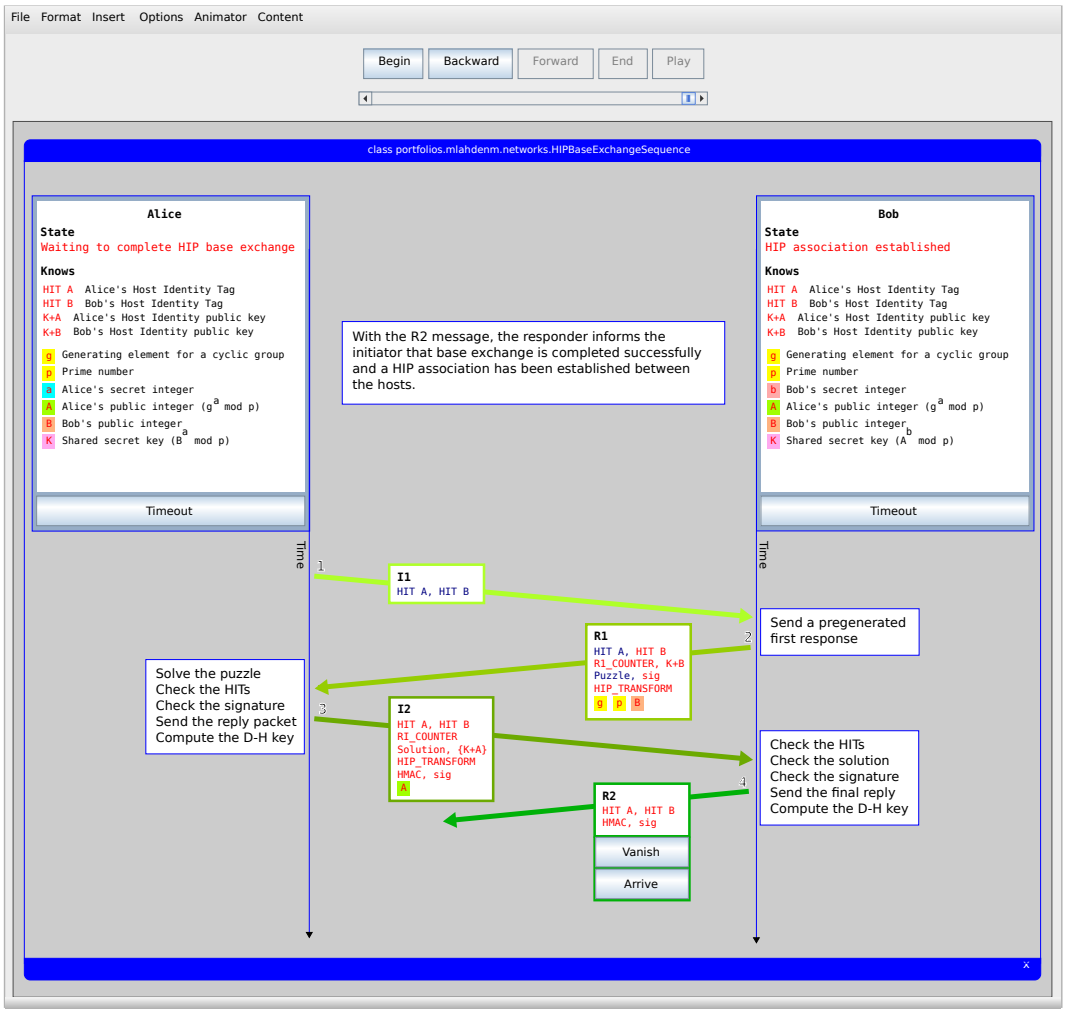

Figure 5.13: Last message in HIP connection establishment.

Bob will now compute the Diffie-Hellman key now shared by Alice and Bob. When the R2 message is sent the following information is shown:

> With the R2 message, the responder informs the initiator that base exchange is completed successfully and a HIP association has been established between the hosts.

When Alice receives the R2 packet she will check its signature and change her state to "HIP association established".

The controls for simulating the Internet, that is deciding whether the messages get to the other host or are lost, are visible in each of the figures.

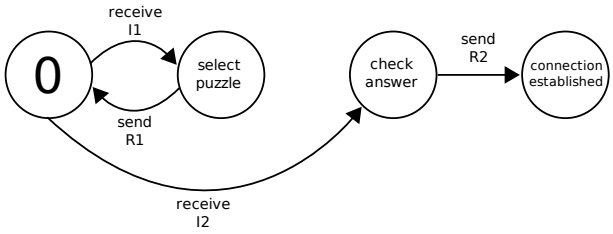

Figure 5.14: Simplified states of the receiver in HIP base exchange.

5.4.2 Mobility and Locator Management

By separating the transport layer from the internetworking layer HIP enables network mobility and host multihoming solutions. HIP messages may contain a LOCATOR parameter to notify peers on alternate addresses for the host. The basic case is presented in Figure 2.3 in which a host adds an address and notifies the peer. The same is visualized in Figure 5.15 and presented next.

In this situation Alice has a mobile host that gets disconnected. Alice then gets a new IP address and sends an UPDATE message to Bob containing the new IP address in the LOCATOR parameter. The LOCATOR parameter also includes a lifetime for the locator. The UPDATE message also includes an ESP_INFO parameter which contains the old and the new Security Parameter Indexes (SPIs) for a security association. In this case they will both be the pre-existing incoming SPI. In addition the message contains the SEQ parameter indicating Bob has to acknowledge the UPDATE. Alice waits for acknowledgement for the UPDATE from Bob and will send the message again if no response is received in a while.

When the first UPDATE message is sent the following is displayed in the info box:

> Upon obtaining a new IP address, the mobile host sends a LOCATOR parameter to the peer host in an UPDATE message. The UPDATE message also contains an ESP_INFO parameter containing the values of the old and new SPIs for a security association. In this case, the OLD SPI and NEW SPI parameters both are set to the value of the preexisting incoming SPI; this ESP_INFO does not trigger a rekeying event but is instead included for possible parameter-inspecting middleboxes on the path. RFC5206 [29]

When Bob gets the above UPDATE message, he validates it and updates bindings between its HIP association and Alice's new destination address. Bob verifies the address by putting a nonce in the ECHO_REQUEST parameter of the update message he will send to the new address. The message also contains the ESP_INFO parameter with both the old and the new SPI values set as the pre-existing incoming SPI. As an acknowledgement Bob will set the ACK parameter to the value of the SEQ parameter in the message received, and in addition request acknowledgement by setting the ACK

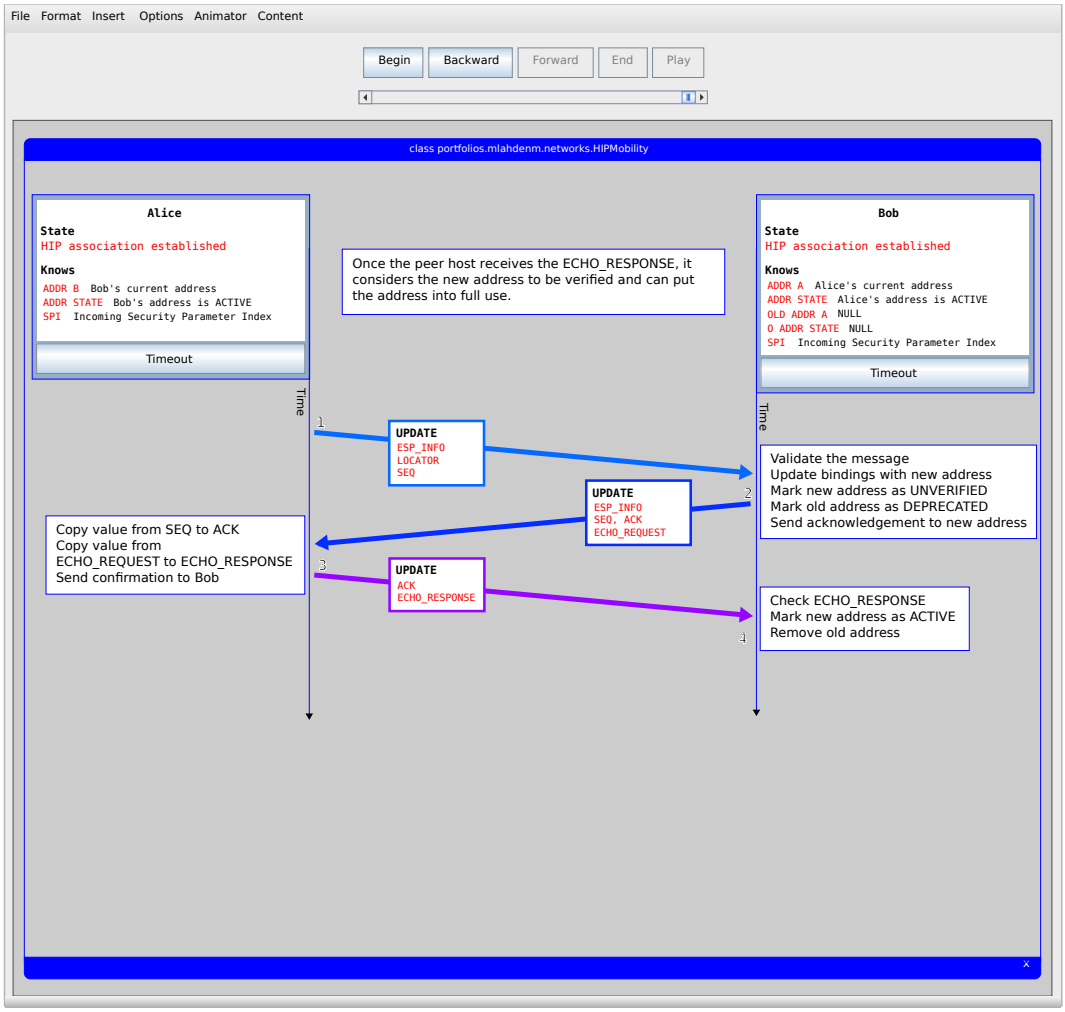

Figure 5.15: HIP mobility.

parameter. Bob will mark Alice's new address as UNVERIFIED and the old address as DEPRECATED.

When the response UPDATE message is sent, the following is shown in the info box:

> The peer host MUST perform an address verification by placing a nonce in the ECHO_REQUEST parameter of the UPDATE message sent back to the mobile host. It also includes an ESP_INFO parameter with the OLD SPI and NEW SPI parameters both set to the value of the pre-existing incoming SPI, and sends this UPDATE (with piggybacked acknowledgment) to the mobile host at its new address. The peer MAY use the new address immediately, but it MUST limit the amount of data it sends to the address until address verification completes. RFC5206 [29]

Once Alice gets the previous UPDATE message from Bob, she will set the ECHO_RESPONSE parameter to the value of the ECHO_REQUEST parameter it received, and the ACK parameter to the value of the SEQ parameter it received.

When Alice sends the UPDATE message in reply, the following is displayed in the info box:

> While the peer host is verifying the new address, the new address is marked as UNVERIFIED in the interim, and the old address is DEPRECATED. Once the peer host has received a correct reply to its UPDATE challenge, it marks the new address as ACTIVE and removes the old address. RFC5206 [29]

When Bob receives the UPDATE message from Alice containing the nonce he sent, he marks the new address as ACTIVE and removes the old address. The following is shown in the information box:

> Once the peer host receives this ECHO_RESPONSE, it considers the new address to be verified and can put the address into full use. RFC5206 [29]

5.4.3 Multihoming

A host may have more than one interface or global address. Lets consider a situation where Alice has established a HIP association through 3G and she wants to use a faster wireless local area network which happens to be available.

A basic multihoming scenario was presented in section 2.2.3 from RFC5205 [29]. Alice will send an UPDATE message with a LOCATOR parameter and the ESP_INFO parameter. She requests a new security association for the new address by setting the

OLD_SPI field of the ESP_INFO parameter to zero and the NEW_SPI to a new value. In the LOCATOR parameter, she will include both the new locator and the new SPI value, and the original address and the SPI associated with it. Alice will also set the Preferred locator field in the new locator's sub-parameter to one, and the that of the original locator to zero. Alice may also send new Diffie-Hellman parameters in the UPDATE message.

When Bob receives the previous UPDATE message, he will set a new SPI value to the ESP_INFO parameter for the UPDATE message he will send, to create a new security association with the new address. He will acknowledge the UPDATE Alice sent with the ACK parameter and verify Alice's new address by a challenge, setting the ECHO_REQUEST parameter to a nonce. Bob sends the response UPDATE to the new address as the UPDATE marked it as the preferred address.

Alice will now receive the acknowledgement UPDATE from Bob with a new outgoing SPI from the ESP_INFO parameter. She will acknowledge this with another UPDATE message with the ECHO_RESPONSE parameter set to the ECHO_REQUEST parameter value of the UPDATE message from Bob.

In the following figure 5.16, Alice has just sent the first UPDATE message. The student has clicked on the LOCATOR parameter of the UPDATE message. This has opened a box showing the sub-parameters NEW_LOCATOR and ORIG_LOCATOR. Clicking the LOCATOR has also brought up new information in the upper information box. Then the user has clicked on the NEW_LOCATOR sub-parameter, which has opened yet another box containing Alice's 2nd address, 2nd inbound SPI, and a boolean value indicating it as the preferred locator.

In Figure 5.17, there is an alternative visualization of the same situation which requires more space and a more intelligent layout engine. Matrix supports the use of different visualization schemes between which the user can change freely.

5.4.4 Man in the middle protection

The visualization also works in situations where there are more than two communicating peers associated with a protocol. Figure 5.18 is a screen shot of a situation where a third party is eavesdropping on the message exchange. As there is less room horizontally, the info box will be displayed on top of the middle party's status box. The picture also displays the focus feature of the visualization, which helps the learner focus on the latest developments in the protocol message exchange by covering other parts of the message exchange temporarily out-of-sight.

In figure 5.19 the status box of the eavesdropper is more visible. It demonstrates how the eavesdropper may acquire the information elements sent in the base exchange but that without knowing the listed information elements he still cannot create the shared key.

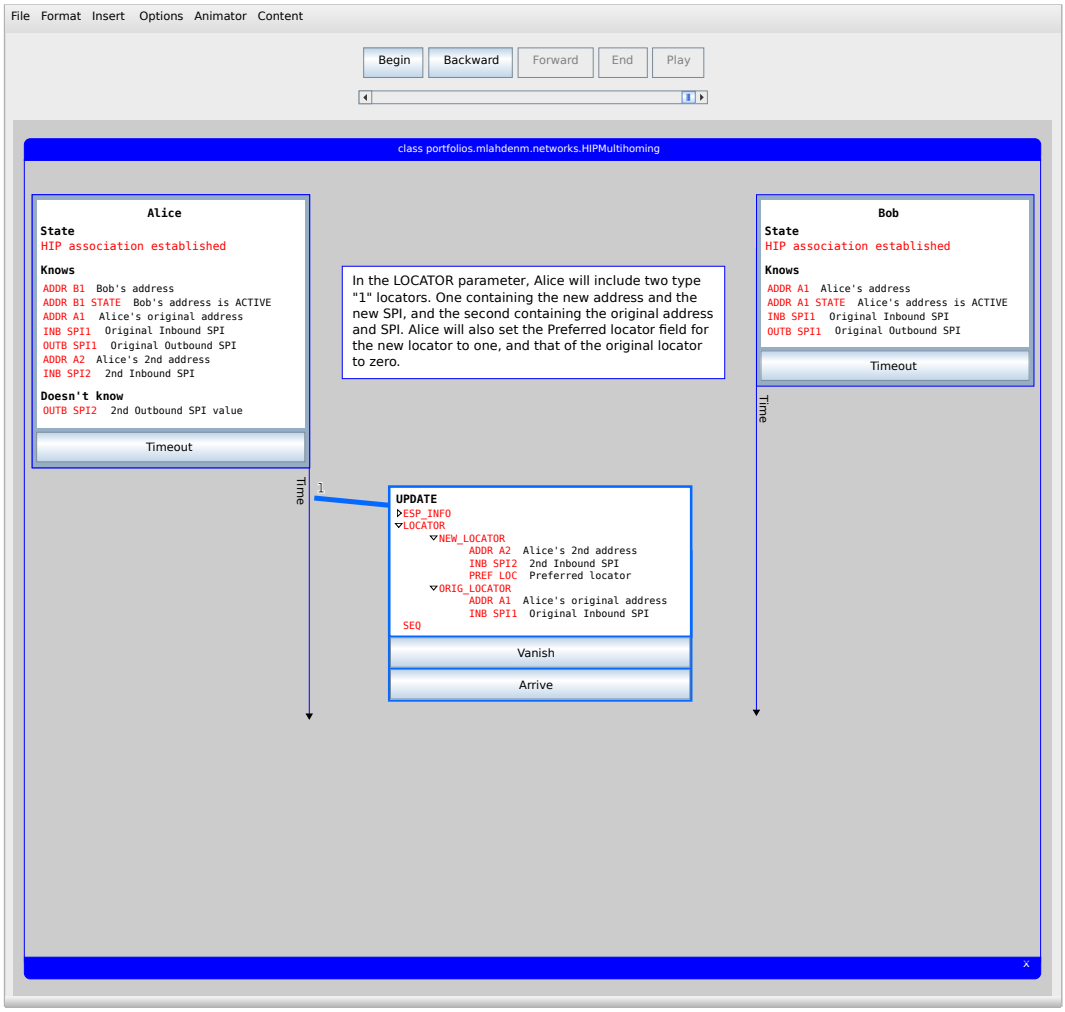

Figure 5.16: HIP multihoming scenario.

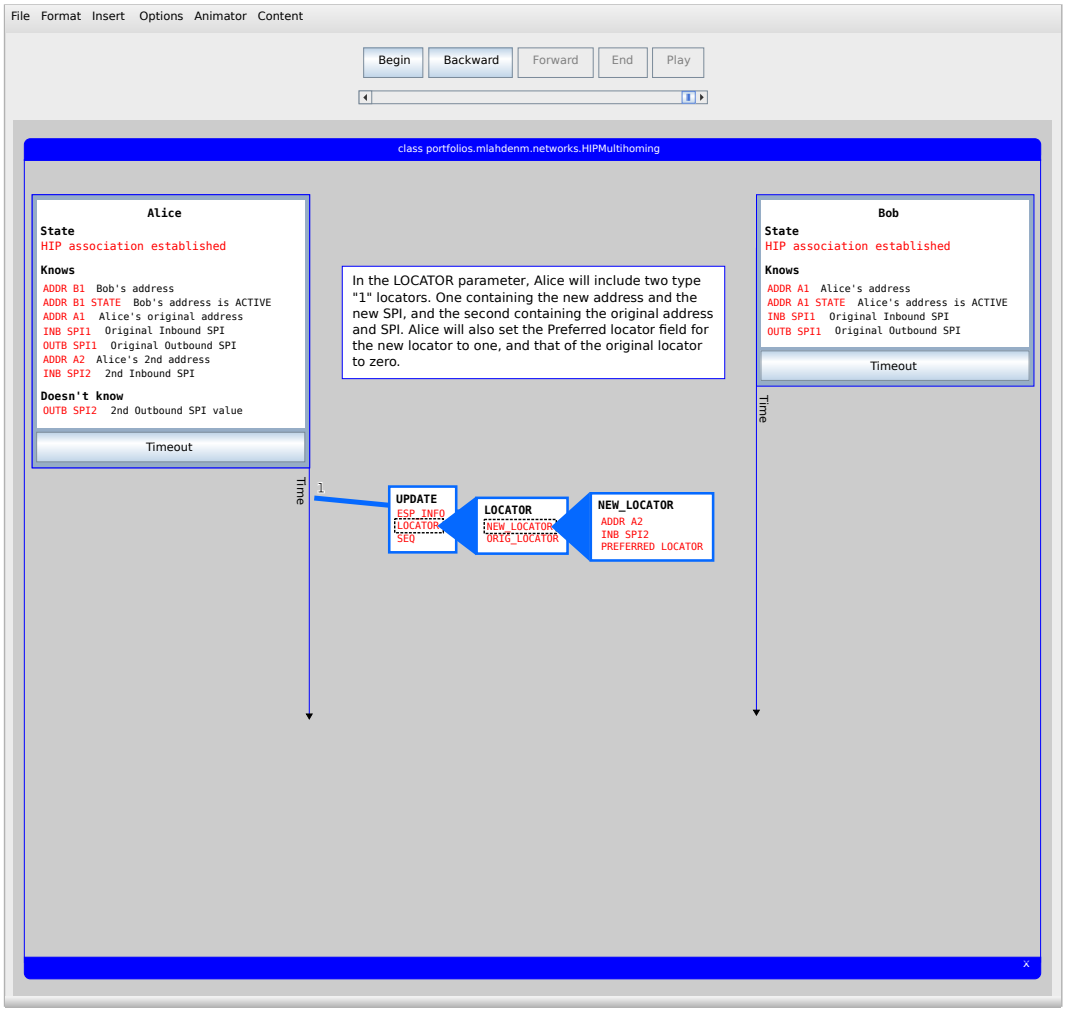

Figure 5.17: HIP multihoming scenario with an alternate visualization scheme.

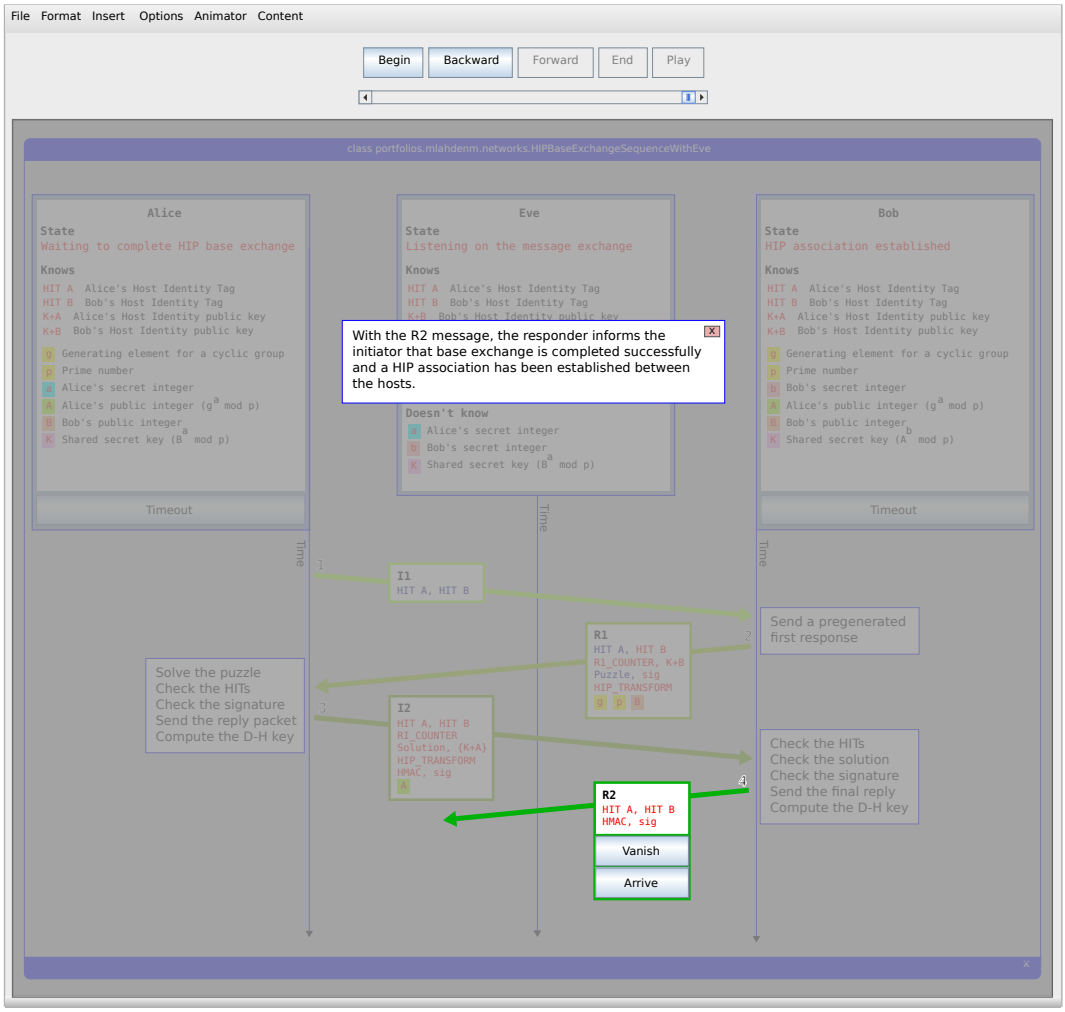

Figure 5.18: HIP base exchange with an eavesdropper listening. An infobox open.

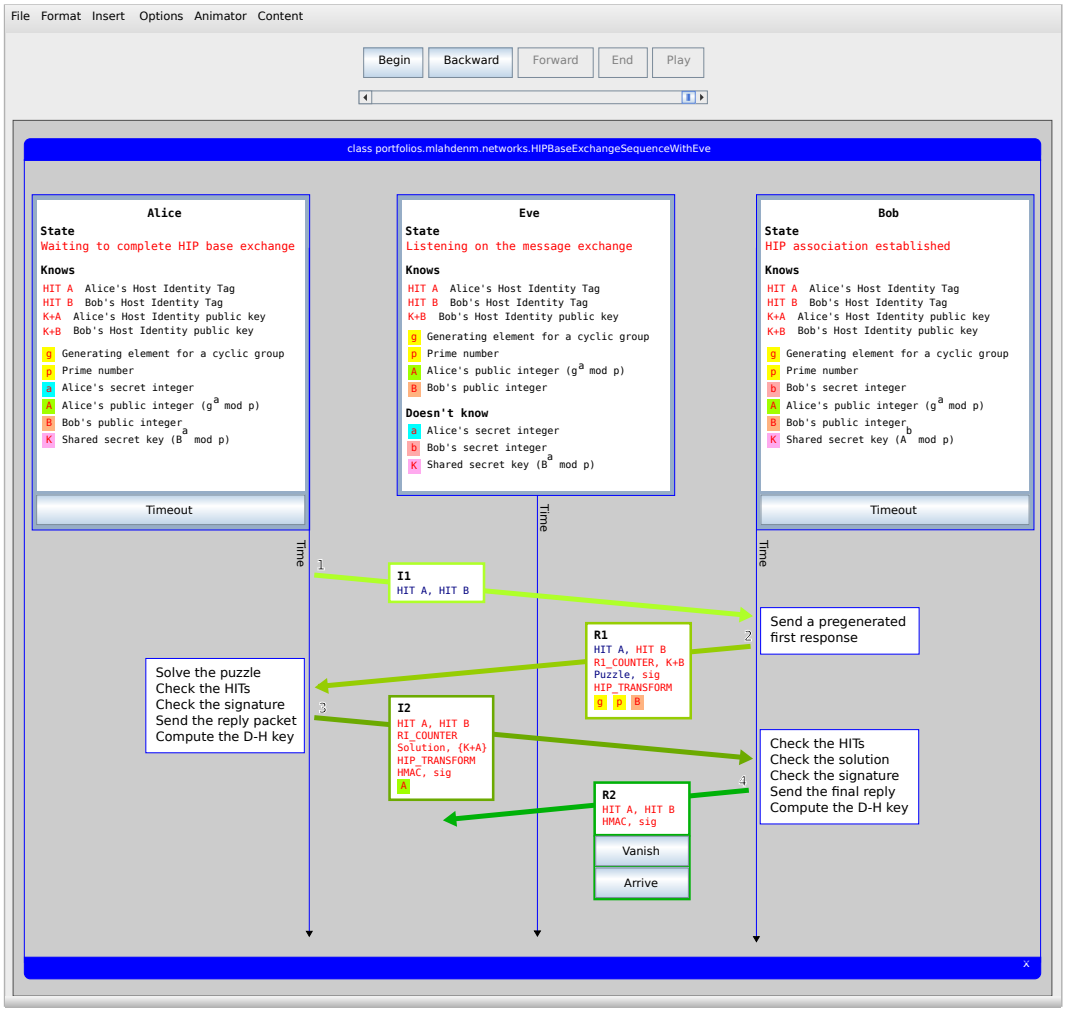

Figure 5.19: HIP base exchange with Eve listening.

5.4.5 Other situations to visualize

Other things to visualize in HIP could be NAT traversal, protection measures against replay attacks, data transfer, and connection termination. In addition to a normal 4-way handshake, there is a possibility for a 3-way handshake and a 2-way handshake, depending on the situation. In visualizing data transfer, one could visualize what happens, for example, in connection problems. There are also several protection mechanisms in HIP against replay attacks which could be visualized, such as

- Pre-generated and pre-signed R2 packets and statelessness protect the Responder against replays of I1 packets.
- The Initiator is protected against R1 replays by a monotonically increasing R1 generation counter.
- The Responder is protected against I2 replays and false I2s by the puzzles which the Responder has given for the Initiator to solve or opaque data which the Initiator must echo.
- Hosts are protected against R2 replays by the use of a less expensive HMAC verification before the HIP signature verification.

Chapter 6

Evaluation

In this chapter I assess how well the solution presented in this thesis answers to the problems it set out to solve. The goal of this thesis was to determine how to create an interactive system to aid in teaching protocols and enhance learning by means of visualization. This involved examining other related work and solutions, and presenting a solution that would address the challenges described in the problem statement in section 1.1. The results of this thesis could not be verified by user evaluation yet, as the implementation of the visualization was not completed during the writing of this thesis. Next is a brief overview of answers provided to the problems in the problem statement followed by a more in-depth evaluation further down.

> Demonstrating Internet principles and protocols is difficult because there is such a diversified set of them.

This is addressed by designing the visualization clear and simple with user interface elements familiar to the student, and creating the visualization general enough to support varied protocols.

> Protocol visualizations are often confined to previously determined protocols and message exchange flows.

The visualization has been designed so that it supports varied protocols, and the architecture enables more protocols to be easily implemented to the system. The flow of the protocol exchange can be steered using visual controls. While the design presented in this thesis suggests presenting a protocol using distinct scenarios, nothing prevents implementing a protocol in a more holistic way, and allowing the user more freedom of control.

> There may be multiple parties involved in message exchange. Depicting this can be hard.

The familiar MSC presentation style allows several hosts to coexist side-by-side. The individually colored messages ease the job of distinguishing the messages from each other. The MSC presentation style is not the best for representing situations with multiple interconnected peers, though. As all the messages sent are left visible, they cover much of the visualization area. A visualization scheme which would only visualize the last few messages would leave more room for positioning hosts optimally. Such a visualization scheme could be implemented for the Matrix platform and selected as an alternative view model.

> Courses often have a massive amount of students and few assistants. There is little possibility for personal guidance, and checking exercises for all the students is a lot of monotonous work.

Building on the matrix platform enables personalized automatically generated and assessed exercises to be created using the TRAKLA2 system.

> In some part connected with the previous, getting feedback may be impossible or students get it so late that learning is hindered.

The automated assessment of exercises provided by the Matrix platform and TRAKLA2 enables immediate feedback.

> Solutions often do not support trying out for yourself.

The visualization was designed to support visual protocol simulation with plenty of more in depth information easily accessible.

6.1 Evaluation criteria

Taking into account the observations made in [27], learner engagement should be carefully implanted to the protocol visualization framework. The Matrix visualization framework, and its derivatives TRAKLA2 and MatrixPro support many of the these. They support responding through automatically generated exercises. Changing is supported in allowing the user to select the input values and control the visualization in many ways. Constructing is implemented in visual simulation. Presenting is supported in many ways. Since exercises created with TRAKLA2 can be individually tailored automatically, this makes it possible to share knowledge between students with less risk of plagiarism.

Common reasons listed by Naps et al in [27] for not using visualizations in education were: it takes too much time to create visualizations, it takes too much time to find and

learn the tools, visualizations take too much time from the lecture, and last, teachers are concerned about the equipment needed in visualizations. By using Matrix, creating visualizations on implemented protocols is as easy as clicking a few buttons. Adding new protocols has been made as easy as possible. Visual elements have been designed so that they can be used to visualize diverse protocols in various situations. Controlling the visualization has been designed straight-forward. Only the necessary controls are visible at one time. By handling different situations in the implemented protocols separately, the amount of detail shown in each situation can be controlled. This makes it easier for a lecturer or a student to use the system. It may also make it easier for a programmer to implement limited parts of a protocol, instead of implementing a full state machine. Of course, implementing e.g. a protocol handshake separately also makes it possible to create situation-specific information, which can mean more work for the designer or implementor.

Creating visualizations for a lecture should be easy and fast. The program needs to be designed and implemented comprehensively and it needs to support common protocols to be worth finding and learning. It should be efficient to use so it does not waste valuable lecture time.

6.2 Generality

A principled taxonomy of software visualization [34] by Price et al proposes a set of categories in which a system may be assessed. Category A.1 is generality.

The solution presented in this thesis strives for generality in that it tries to support the visualization of diverse protocols by using general visualization elements and a layout that even supports displaying several hosts or peers at the same time. It also supports the visualization of concurrent actions as well as visualizing several protocols side by side, although using manual synchronization. It is, though, confined to visualizing protocols that have been implemented to the system.

Adding new protocols to the system requires programming skills, but should be easy by reusing previous implementations. In addition, the solution proposes dividing the visualization of a protocol to distinct situations instead of having one complete implementation of a protocol. This limits the visualization of a protocol to the implemented situations. This can be a constraint on generality, but the design decision was made to support teaching and studying by enabling the generation of more situation specific customizations and adjusting the visualization to the level of understanding of the learners.

6.3 Content

The visualization presented in this thesis does not visualize the code or an actual implementation of a protocol. The focus is more in visualizing the high level protocol, a distributed algorithm. Control flow and data flow are primary subjects for visualization in the solution. The visualization concentrates on the message exchange and state information of protocols. The user is usually in control of the actions the protocol takes, such as the sending of messages and timeouts. A host may also be automated. When displaying a specific situation of a protocol, the buttons controlling unrelated actions are hidden. The level of detail displayed can be adjusted for separate use cases and situations. For example, only a set of the most basic elements may be displayed to clarify basic functionings. This will also leave more room for using broader explanations for the remaining elements. For a more advanced example, the message contents may be visualized in a tree structure to correspond information in an actual packet.

Fidelity and completeness in this visualization relate to whether the visual metaphors used present the true and complete behavior of the protocols. Although one can implement a protocol visualization using the visual elements presented in this thesis to display all the information involved in protocol packet exchange, it is useful to leave out some of the non-essential information so it does not distract the learner from more essential information. The purpose is to teach the basic functionality of a protocol, not every little detail required to implement a protocol. Also, it may not be purposeful to visualize actual figures from actual protocol message exchange, but rather display shorter and more clear figures or metaphors. So fidelity may not always be a top priority in pedagogical use. Additional informational elements are also used in the solution presented in this thesis.

6.4 Form

The overall layout of the visualization is similar to a message sequence chart (MSC). It is a common tool for visualizing interactions in real-time systems, and familiar to most information science students. The graphical vocabulary used in my visualization is composed of simple graphical elements. A host is visualized as a rectangle containing a name for the host, state information, and a list of known information elements. Timelines attached to the hosts point directly down representing that newer events appear lower than older ones. Messages or packets are visualized as slightly downward arrows which have a box attached to them containing the possible name for the packet, and a list of information elements visualizing the information fields of the packet.

Colors are used to help distinguish messages from each other. A different color scheme is used for messages connected to different phases of protocol message exchange.

In addition to these elements, the visualization uses control buttons and a slide bar

representing the temporal aspect. The similarity to message sequence charts helps students in learning this new visualization, and the simple presentation style enables the visualization to be used for varied protocols. Not all the elements have to be visible all the time. Instead one can use visualization schemes for controlling how much detail is shown. Coarse-grain visualization is especially useful when learning about new things. The system also support eliding information. That is to hide parts of the information which are not of immediate interest.

6.5 Method

Using the Matrix platform allows visualizations to be created and controlled completely visually once a protocol is implemented to the system. New general or even protocol specific visualizations may be created by implementing interfaces. The visualization is highly tailorable. The user may select between visualization schemes if more than one have been implemented for the situation.

6.6 Interaction

The user interface of the design follows the style of a typical window-based program. Different situations for a protocol may be selected from a menu. On-screen buttons give control over the hosts and the network e.g. losing data packets. In addition many information elements may be clicked to view a general description on them. This follows the hyper-text style most familiar to students nowadays.

In addition the system supports timeline controls. The visualization may be navigated in using the time line controls and scroll bars. The user may freely traverse in both temporal directions using the timeline controls. As the history of the message exchange is left visible, this only serves to make it even more easier to examine the message exchange. Packet data is presented as a tree whose subtrees may be hidden or displayed providing elision control.

6.7 Effectiveness

To evaluate the effectiveness of the visualization, first, we need to state the purpose of this visualization. One of the purposes of this visualization is to support classroom demonstration, both novice and advanced level. A second purpose for the visualization is for it to enhance self-studying, and in connection to that, support doing exercises, and assist in communicating knowledge to others. The system supports different levels

of learner engagement: viewing, responding, changing, constructing, and presenting. A possible purpose for the visualization would also be to use it in designing protocols.

To assess the appropriateness and clarity of the visualization the system properly the system would need to be implemented e.g. for a course in data communications. Things to evaluate would be the ease of implementing new protocols, ease and effectiveness of using the system on a lecture, and effectiveness of self-studying using the system. As it is the design can be assessed by analytical means such as a cognitive walkthrough, much like what has been done in chapter 5, and utilizing a heuristic evaluation. Jakob Nielsen has listed ten general principles for user interface design in [28]:

Visibility of system status "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time."

As the solution presented is not real-time, the system usually expecting user input to continue. State information for the associated hosts is visible at all times.

Match between system and the real world "The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order."

Language in the state information, information elements, and descriptions has been written using familiar concepts, and providing links to descriptions for new concepts.

User control and freedom "Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo."

The time line controls enabled by the matrix platform fully support undoing and redoing.

Consistency and standards "Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions."

The visualization has been designed to be general enough to support varied protocols using the same controls. The controls follow quite common conventions, like: buttons, scrollbars, menus, and hyper-links.

Error prevention "Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone con-

ditions or check for them and present users with a confirmation option before they commit to the action."

Non-allowed actions are hidden from sight so the user can not press them or be confused about them.

Recognition rather than recall "Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate."

All the allowed actions are visible at any one time. Using TRAKLA2, instructions for an exercise are always visible on the top of the page.

Flexibility and efficiency of use "Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions."

Support for accelerators has not yet been planned.

Aesthetic and minimalist design "Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility."

The solution displays a simple design and supports information elision.

Help users recognize, diagnose, and recover from errors "Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution."

Error messages had not yet been designed.

Help and documentation "Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large."

Many of the information elements are displayed as links, and clicking the link will show more information concerning them.

6.8 Personal evaluation

Assessing my work, I feel that I have succeeded in creating a design for protocol visualization that is general and flexible. The user interface conforms to usability guidelines. Also, the Matrix platform seems ideal to build upon. Visualizing protocols using scenarios enables focusing on essential parts which enhances learning.

But, not being able to finish the implementation also meant that no user evaluation could be executed. The thesis was, almost entirely, limited to HIP visualization examples. Having example visualizations of other protocols as well might have pointed out omissions in the architecture.

All in all, the work I have done so far seems like a good basis for creating a software visualization for protocols. The work is not yet fully done, and should be continued by actually implementing the design.

Chapter 7

Conclusions and Future Work

This thesis work presents a design for visualizing data communications protocols on top of the Matrix framework that is originally designed for algorithm visualizations. A pilot version testing out protocol visualization for Matrix was used as the general starting point. Ideas were gathered from related visualization articles and implementations, and the best were integrated to the design. As the implementation was not finished during the writing of the thesis, the design was evaluated using a cognitive walkthrough.

The visualization presented in this thesis addresses the defined requirements: It is general enough to support varied protocols and scenarios. The flow of the scenarios can easily be steered using visual controls. The visualization supports multiple interconnected peers, and features, such as support for on the fly presentations, automatic assessing of exercises, and self study, are enabled by the Matrix platform.

Dividing the visualization of a protocol into smaller pieces helps learners concentrate on the essential in different situations. Of course, choosing the most important features and concentrating on them is essential. Different use scenarios have different needs. The level of detail shown, and deciding which elements to display, depends on the situation. Too much detail can cloud the general idea. Students on basic courses have different needs than those on more advanced courses.

The design in this work enables lots of information to be included to a visualization to describe different aspects and elements of the protocol message exchange. These include descriptions for, e.g., the information elements involved, the different phases in the protocol message exchange, and the actions performed by the hosts involved. Different visualization schemes may be implemented which present the message exchange on different detail levels. The teacher or the student may switch between different visualization schemes as necessary. Implementing new protocols requires a fair amount of work with deciding which parts of a protocol to focus on, planning the different scenarios, and inputting all the information. But after that initial work is done, creating presentations and self studying is easy.

There is still much to be done related to the work done in this thesis. Designing, at least a few, visualization scenarios for other protocols, such as TCP, UDP, and SMTP, would be a good next step. After this, the visualization should be implemented to the Matrix platform, MatrixPro, and TRAKLA. It could then be put to test on the course Introduction to data communications and multimedia. User experiences should be gathered on the course for evaluating the usefulness and effectiveness of the system. The ease of creating class demonstrations should be evaluated. The use of the visualization for self study should be evaluated. Some exercises could also be executed utilizing the automatic assessment capabilities of TRAKLA, and the experiences recorded for analyzing.

With the experiences and feedback from the course, the system could be further developed. More protocols could be implemented, such as ICMP, FTP, HTTP, POP3, SSH, and Mobile IP. Each protocols have distinguishing features, e.g. concerning mobility and security, whose visualization requires thought from the teacher, as well as from the student. After implementing more protocols, the software could be used on different courses on data communications and network security.

Bibliography

[1] ALLMAN, M., PAXSON, V., AND STEVENS, W. TCP Congestion Control. Tech. Rep. 2581, Apr. 1999. Obsoleted by RFC 5681, updated by RFC 3390.

[2] BAECKER, R. M. Sorting out sorting. Narrated colour videotape, 30 minutes, now available at http://www.youtube.com/, 1981.

[3] BAECKER, R. M. Sorting out sorting: A case study of software visualization for teaching computer science. In *Software Visualization: Programming as a Multimedia Experience*, M. Brown, J. Domingue, B. Price, and J. Stasko, Eds. The MIT Press, Cambridge, MA, 1998, ch. 24, pp. 369–381.

[4] BROWN, M. H., AND SEDGEWICK, R. Techniques for algorithm animation. *IEEE Software 2*, 1 (January 1985), 28–39.

[5] COLE, L. *A history of education: Socrates to Montessori*. Rinehart, 1950.

[6] COMBS, G. Wireshark 1.2.7. GNU General Public License version 2. Software available at http://www.wireshark.org/, 2010.

[7] ELMQVIST, N. Protoviz: A simple security protocol visualization. Report posted at http://www.cs.chalmers.se/ elm/courses/security/report.pdf, 2004.

[8] FURCY, D., NAPS, T., AND WENTWORTH, J. Sorting out sorting: the sequel. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education* (New York, NY, USA, 2008), ACM, pp. 174–178.

[9] HNATYSHIN, V. Y., AND LOBO, A. F. Undergraduate data communications and networking projects using opnet and wireshark software. *SIGCSE Bull. 40*, 1 (2008), 241–245.

[10] HUNDHAUSEN, C. D. *Toward Effective Algorithm Visualization Artifacts: Designing for Participation and Communication in an Undergraduate Algorithms Course*. Unpublished doctoral dissertation (tech rep. no. cis-tr-99-07), University of Oregon, 1999.

[11] HUNDHAUSEN, C. D., DOUGLAS, S. A., AND STASKO, J. T. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing 13*, 3 (June 2002), 259–290.

[12] ITU-TS. ITU-TS Recommendation Z.120: Message Sequence Chart 2004. Tech. rep., ITU-TS, Geneva, 2004.

[13] JACOBSON, V., BRADEN, R., AND BORMAN, D. RFC 1323: TCP Extensions for High Performance. Tech. Rep. 1323, IETF, May 1992.

[14] JOKELA, P., MOSKOWITZ, R., AND NIKANDER, P. Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP). Tech. Rep. 5202, Apr. 2008.

[15] JUNCO, R., AND MASTRODICASA, J. *Connecting to the net.generation: What higher education professionals need to know about today's students*. NASPA, 2007.

[16] KARAVIRTA, V., KORHONEN, A., AND MALMI, L. MatrixPro. Computer program, November 2003.

[17] KARAVIRTA, V., KORHONEN, A., MALMI, L., AND STÅLNACKE, K. MatrixPro - a tool for demonstrating data structures and algorithms ex tempore. In *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies* (Joensuu, Finland, 2004), pp. 892–893.

[18] KOIVIKKO, U., AND LÃŰNNBERG, J. Visualization for teaching automatic repeat request. Report posted at http://www.cs.hut.fi/ jlonnber/T-106.5800/ackvis-final.pdf, 2007.

[19] KORHONEN, A. Visuaalinen algoritmisimulaatio ja sen sovelluksia. *Tietojenkäsittelytiede*, 23 (2005), 42–59.

[20] KORHONEN, A., AND MALMI, L. Algorithm simulation with automatic assessment. In *Proceedings of The 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'00* (Helsinki, Finland, 2000), ACM Press, New York, pp. 160–163.

[21] KORHONEN, A., MALMI, L., AND SILVASTI, P. TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. In *Proceedings of Kolin Kolistelut / Koli Calling – Third Annual Baltic Conference on Computer Science Education* (Joensuu, Finland, 2003), pp. 48–56.

[22] KORHONEN, A., MALMI, L., SILVASTI, P., KARAVIRTA, V., LÖNNBERG, J., NIKANDER, J., STÅLNACKE, K., AND IHANTOLA, P. Matrix - a framework

for interactive software visualization. Research Report TKO-B 154/04, Laboratory of Information Processing Science, Department of Computer Science and Engineering, Helsinki University of Technology, Finland, 2004.

[23] MATHIS, M., MAHDAVI, J., FLOYD, S., AND ROMANOW, A. RFC 2018: TCP Selective Acknowledgment Options. Tech. Rep. 2018, IETF, Oct. 1996.

[24] MATOCHA, J., CAMP, T., AND HOOPER, R. Extended analogy: an alternative lecture method. *SIGCSE Bull. 30*, 1 (1998), 262–266.

[25] MOSKOWITZ, R., AND NIKANDER, P. Host Identity Protocol (HIP) Architecture. Tech. Rep. 4423, May 2006.

[26] MOSKOWITZ, R., NIKANDER, P., JOKELA, P., AND HENDERSON, T. Host Identity Protocol. Tech. Rep. 5201, IETF, Apr. 2008.

[27] NAPS, T. L., RÖSSLING, G., ALMSTRUM, V., DANN, W., FLEISCHER, R., HUNDHAUSEN, C., KORHONEN, A., MALMI, L., MCNALLY, M., RODGER, S., AND ÁNGEL VELÁZQUEZ-ITURBIDE, J. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin 35*, 2 (June 2003), 131–152.

[28] NIELSEN, J. Enhancing the explanatory power of usability heuristics. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1994), ACM, pp. 152–158.

[29] NIKANDER, P., HENDERSON, T., VOGT, C., AND ARKKO, J. End-Host Mobility and Multihoming with the Host Identity Protocol. Tech. Rep. 5206, Apr. 2008.

[30] OECHSLE, R., GRONZ, O., AND SCHÜLER, M. Visusniff: A tool for the visualization of network traffic. In *Proceedings of the Second Program Visualization Workshop* (2002), pp. 118–124.

[31] POSTEL, J. Internet Protocol. Tech. Rep. 791, Sept. 1981. Updated by RFC 1349.

[32] POSTEL, J. RFC 793: Transmission Control Protocol. Tech. Rep. 793, IETF, Sept. 1981. Updated by RFCs 1122, 3168.

[33] PRATCHETT, R. Gamers in the UK: Digital play, digital lifestyles. White paper posted at http://open.bbc.co.uk/newmediaresearch/files/BBC_UK_Games_Research_2005.pdf, 2005.

[34] PRICE, B. A., BAECKER, R. M., AND SMALL, I. S. A principled taxonomy of software visualization. *Journal of Visual Languages and Computing 4*, 3 (1993), 211–266.

[35] SCHWEITZER, D. L., III, L. C. B., COLLINS, M. D., BROWN, W. C., AND SHERMAN, M. Grasp: A visualization tool for teaching security protocols. In *Proceedings of the 10th Colloquium for Information Systems Security Education* (June 2006).

[36] SHAFFER, C. A., COOPER, M., AND EDWARDS, S. H. Algorithm visualization: a report on the state of the field. *SIGCSE Bull. 39*, 1 (2007), 150–154.