Outsourcing of Spam Filtering

- with Emphasis on Trust Management

Allan Larsen
Søren Meyer

4th June 2003

The Faculty of Engineering and Science

Department of Computer Science

TITLE: Outsourcing of Spam Filtering
- with Emphasis on Trust Management

PROJECT PERIOD:
Master Thesis (DAT6),
4th of February 2003 - 4th of June, 2003

PROJECT GROUP:
E1-119

GROUP MEMBERS:
Allan Larsen
Søren Meyer

PROJECT SUPERVISOR:
Finn V. Jensen

NUMBER OF COPIES: 6

TOTAL PAGENUMBERS: 76

SYNOPSIS:

In this report the objective is outsourcing of spam filtering to other machines in order to save resources. Doing this is accomplished by using trust management and Bayesian networks. Trust management is analyzed in order to model aspects regarding trust in the Bayesian networks. A system, TrustOne, making outsourcing of spam filtering possible is designed, in which decision making is handled by Bayesian networks incorporated with trust management. Thus being able to handle risks and specify the confidence required to trust a machine to do spam filtering on the behalf of another machine. The resulting system encircle the source of the spam with machines running spam filters, by using outsourcing of spam filtering. This results in less network bandwidth usage and less resources used on machines in order to run spam filter.

Summary

This report proposes a solution called TrustOne, to some of the problems caused by the huge amounts of spam on the Internet. The main focus is to reduce the increasing amount of resources wasted on spam.

The main idea of the solution is to outsource spam filtering to machines closer to the sources of the spam. By removing the spam before it reaches its intended destination, the resources normally consumed by delivering the spam can be saved.

Most people are very sensitive about their e-mail. They expect that when an e-mail is sent, it will reach the intended destination. Thus before a spam filter should be allowed to remove e-mails that are classified as spam, we need to be confident that the spam filter is capable of performing the task to our satisfaction.

Trust and confidence is therefore at the very core of TrustOne. As a result, the basic concepts of trust management is introduced. The trust management techniques described uses evaluations and assumptions. As spam filtering is an automated process, a more mathematical approach is required. Thus methods for converting the results of the trust management process into Bayesian networks are developed and described.

After a brief description of the structure of the Internet and the internal structure of e-mails, the first stage of TrustOne is developed. This simple stage is the simplest possible version of TrustOne, using a very simple communication protocol between the machines. It will remove spam close to its source thus saving resources. Unfortunately this simple design relies on some undesirable assumptions in order for it to work. A second more advanced stage of TrustOne is developed.

In the second stage a more advanced communication protocol is developed, which allows the machines to communicate more efficiently and also removes some of the undesirable assumptions previously required. It is also possible for the end users to retain some of the control of the filtering normally lost by server-side spam filters. The two main decisions of TrustOne: “Do I want to filter someone else’s e-mails”, and “Do I allow someone else to filter my e-mails” are modeled as two Bayesian networks. At this stage TrustOne still assumes that all information communicated between machines is 100% correct. This assumption is undesirable

and thus a trust management process is used to analyze the situation.

In the third and final stage, the result of the trust management process is then used to improve and extend the two Bayesian networks from the second stage. This improvement removes the assumption of 100% correct information and makes TrustOne able to save even more resources.

To test TrustOne, a piece of software is designed and implemented, which is capable of simulating TrustOne on a small network of 15 machines. With this software several tests are performed on the behavior of the communication protocol of TrustOne. It is verified that TrustOne will indeed outsource the filtering to machines close to the source of the spam.

The two Bayesian networks are tested by inserting evidence and observing the results.

As a last confirmation of TrustOne's behavior, both the communication protocol and the Bayesian networks are tested working in unison. The results show that even if some machines are unable/unwilling to use TrustOne, then TrustOne still achieves the desirable result of removing spam close to the source of the spam.

The last tests examine just how resource saving TrustOne is. The results show that a significant amount of resources can be saved when compared to scenarios where TrustOne is not used. There is a drawback though, TrustOne requires that a few machines must spend a lot of resources in order for other machines to be able to save their resources.

As a conclusion, TrustOne seem capable of saving a lot of machines quite a lot of resources, while still allowing the end users some control of the filtering process.

Preface

This report is the documentation of a Master thesis made at Aalborg University, Department of Computer Science, Decision Support Systems. The duration of the project was from February 4th 2003 to June 4th 2003. The purpose of the project was to develop a system capable of reducing the resources wasted on spam by outsourcing of spam filtering. This has been achieved by combining elements of trust management and Bayesian networks into a communication protocol which allow machines to communicate spam related information between each other.

A CD-ROM containing the Bayesian networks and source code used throughout this project is located at the inside of the back cover. A ReadMe file on the CD-ROM describes the content of the disk.

Allan Larsen

Søren Meyer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

What is spam? Some define spam as: 'unsolicited commercial e-mail', while others define it as: 'unsolicited automated e-mail'. Spam is sort of the computerized version of the ads almost everyone receive in their mailbox. Traditionally the post delivery office has delivered the ads to each household, but as more and more people now have access to the Internet and are able to receive e-mails, spamming has become a very cheap way of advertising. It is possible to send literally millions of e-mails advertising some product at almost no cost. Most of the receivers of these e-mails will never read them, and even if they do, they will most likely never take advantage of the offer. Given that the advertising is almost free, only a very small percentage of the receivers actually have to buy the product in order to cover the advertisement cost. When compared to the expenses of printing and delivering millions of paper ads, 'spamming' can be a very attractive alternative for companies with something to sell.

No matter how spam is defined, most people agree that spam is something that they do not want. They did not ask for the e-mails, but they receive them anyway. The problem is not that a spam message is received, but rather that so many are received. It is not uncommon for a single person to receive 30 or more spam e-mails each day. Most people find the task of finding the e-mails they want between all the spam time consuming and irritating. Many also consider the act of spamming as bad taste and a questionable way of conducting business. Most companies respect this attitude and only send advertising e-mails to the people who request it. Unfortunally many of the more questionably businesses like pornography web sites, people with 'The Ultimate Recipe For Getting Rich Quick' schemes, and others trying to scam you out of your money, are using spam as their main source of advertisement.

What is a spam filter? Spam filters are designed to remove the spam e-mails

before the receiver sees them. Thus a spam filter is just an automated version of the filtering and classification of e-mails we all do when looking through our e-mails. Even though almost all people find it easy to correctly classify an e-mail as spam, creating an automated spam filter, is not that simple a task. Humans understand the meaning of the words and sentences in the e-mails, the spam filters do not have that luxury.

Also some people do not mind receiving some e-mails that are spam according to the definition of spam. For instance, when buying something from an Internet web-shop, sometimes the web-shop keeps sending e-mails advertising for their products. These e-mails are spam according to the definition of spam, but some people like these e-mails. Thus today, most of the spam filters can be configured to the desires of the individual person using the filter.

Even if a spam filter has found a good way of identifying spam from non-spam, the people sending the spam are actively seeking ways around the spam filters. They do this by altering the structure of the spam e-mails. The spam filter must now find ways to identify these new structures, but the spammers will again change the structure and so on.

Another problem is trust. Just as you trust the post office to safely deliver a letter from one place to another, you need to be able to trust that a sent e-mail will arrive at its destination and actually read by the intended person. A spam filter that classify a non-spam e-mail as spam and removes it, can do serious harm. Important messages might never reach the intended person. Thus we need to have a sufficient level of confidence that the spam filter does its job correctly. Identifying the required confidence in a spam filter or a machine running spam filter is no simple task, since many unknown factors are involved. It is all about being able to analyze the risks being posed and the agents that need to be trusted. A trust management process would help analyzing this, thus improving the task of identifying the required confidence. Trust management is required in order to analyze a given situation thoroughly resulting in making the decision, whether or not to trust a spam filter or the machine running it, manageable.

Some spam filters do a pretty good job of separating spam from non-spam, but they only do so when the e-mail arrives at its destination. It is the same as hiring a person to remove all the unwanted ads from your mailbox, just before you empty it. A lot of resources have still been used on creating the ad and delivering it to your mailbox. These resources have effectively been wasted. Spam produces much the same problem. The spam e-mail might have traveled through many routers on its way to its destination using up computer resources and network bandwidth, only to be removed by a spam filter when reaching its destination. Even though the resources needed by a mail server to process a single e-mail of moderate size is small, the sheer amount of spam e-mail sent each day will tie up a lot of resources.

The amount of spam is growing rapidly. Currently, 50% of all e-mail traffic on the Internet is considered spam. This percentage is expected to rise to 70% by the year 2007 if no significant changes are made to the way e-mails work[13].

1.1 Problem Domain

We will design a server-side spam filtering system, which will be called *TrustOne* that is able to identify spam before the it reaches its final destination, thereby reducing the amount of resources currently used on delivering the spam to its destination. The closer the spam can be identified to its source (the spammer), and thus removed, the less resources would be wasted.

The main problem we will focus on in this report is trust. By removing the spam before the destination, the receiver loses control of the spam filter filtering his e-mail. We therefore need to have confidence that a spam filter which is operating outside our control, will be able to correctly identify and remove only what we consider spam and nothing more.

This will be solved by taking advantage of trust management incorporated in Bayesian networks. Doing this requires an analysis of how knowledge identified in a trust management process can be modeled in a Bayesian network.

TrustOne will be developed in stages with increasing complexity. The initial stages will develop the general functionality of TrustOne. Trust management will be introduced in the third and final stage.

There are many other interesting problems that need to be addressed in order for such a system to be able to function in the real world. Although we will not deal with these problems in detail in this report we will just mention a couple here:

When sending information through the Internet, only the source and destination are 'normal' computers. Routers and router protocols handle the actual transportation. These routers will have to be extended in such a way that they are able to scan for spam and handle the trust management needed by TrustOne.

The way routers scan for and identify spam is critical for success of TrustOne. If the routers do a poor job at identifying spam, the trust management in TrustOne will fail, and everyone can only trust themselves to do the spam filtering properly.

Chapter 2

Decision Making & Trust Management

Whenever decisions are to be made they are considered by evaluating the possible risks of taking the action and the associated likelihood that the risk will be realized. This evaluation process takes place automatically in our heads whenever some decision has to be made. However, sometimes it is simply not possible to make that evaluation. The situation can be complex or contain many unknown factors that at first seem impossible to handle or evaluate. One complex decision to make is to determine if information or behavior of some source is reliable and trustworthy. How do we measure whether or not someone is to be trusted? A mechanism capable of dividing the problem into manageable parts is required. These individual parts can then be analyzed thoroughly and thus help establishing the trust needed.

Povey[4] proposes a process encompassing the above mentioned needs. This process is called a trust management process and the result of this process is a trust policy. Throughout this chapter this trust management process will be described and analyzed with the objective of being able to incorporate trust management into Bayesian networks[1].

2.1 The Trust Management Process

A trust management process will greatly enhance the procedure of evaluating a scenario in order to make the right decision of whether or not to trust a source. In the following sections aspects of trust management will be examined thus giving a deeper understanding of what makes the trust management process advantageous.

[1] Although we are dealing with influence diagrams throughout this report we will mention them as Bayesian networks.

Information regarding risk and trust management throughout this chapter is based on [4] and [8]. A trust management process can contribute with an analysis of possible assets that could be harmed when a decision of trusting some source has been made. These assets should be protected. Thus analyzing what can harm them and the likelihood of them being harmed will be a key aspect of the trust management process.

A trust management process will be useful whenever someone should be trusted, for instance when making investments, or even simple decisions like whether or not to take the train. Taking the train is also an evaluation of risks and whether or not to accept these risks, but it is quite simple and should be decidable without taking advantage of a trust management process. In more complex situations the decision whether or not to trust a source would be eased by analyzing the situation according to the trust management process.

The trust management process based on [4] contains the following main steps:

1. Identify contexts (see Section 2.1.1) and entities (see Section 2.1.2).

2. Identify assets including the threat to these assets and the expected impact should the assets be harmed. (See Section 2.1.3.)

3. Calculate the expected utility of trusting the entities.

4. Identify the vulnerabilities which can lead to the identified threats arising. (See Section 2.1.4.)

5. Analyze the risks of the vulnerabilities by considering likelihood and consequences. (See Section 2.1.4.)

6. Determine the required trusting beliefs and confidences required to trust/distrust an entity. (See Section 2.1.5.)

7. Identify metrics (see Section 2.1.7.) and confidence in the accuracy of these.

8. Relate the cost of the metrics to the expected utility and their contribution to the confidence in the trusting beliefs. Select a subset of metrics which should be used in each context.

9. Using the metrics, establish the beliefs identified and determine whether the required confidence levels are encompassed.

10. Based on this evidence; accept, reject or treat the risk and reevaluate.

2.1.1 Contexts

The trust contexts is simply the different scenarios in which a decision has to be made, for instance whether to perform an action given that some entity can be trusted or distrusted.

2.1.2 Entities

An entity can be seen as the agent that is to be trusted. Entities can for instance be:

- A person
- A system
- An organization
- A company
- A computer program/system/technology/protocol etc.

Some entities might be very well known in which case no trust management process would be required. In case the confidence in a given entity is unknown, or simply not definable, a trust management process would ease the decision making.

2.1.3 Assets

Assets are things that should be protected. Any harm against these is not desirable and should be avoided. Assets could for instance be money, resources or some private information. In the trust management process the threats to these assets are identified and the impact they might have is evaluated.

2.1.4 Vulnerabilities and Risks

What makes the assets vulnerable should be determined. The vulnerabilities help specify weaknesses, which can lead to threats being realized thus harming our assets. Vulnerabilities can lead to risks which should be analyzed. Risks are analyzed by evaluating their likelihood and consequence. In other words to specify whether or not to accept a risk one should consider how likely the threat is to be realized and the impact it will have if realized. The point at which a risk is acceptable is a balance between likelihood and the consequence if the threat is realized. This balance can be illustrated as:

$$P(harm) \times cost(harm) \leq \epsilon \tag{2.1}$$

where $P(harm)$ is the probability (likelihood) of the risk being realized (thus harming our assets), and $cost(harm)$ is the impact/cost/consequence the threat posed by the risk might have if realized. ϵ can be seen as a threshold and determines whether or not to run the risk. Only if $P(harm) \times cost(harm)$ is less than some specified value of ϵ should the risk be accepted. The value of ϵ should be determined carefully considering the given situation. When considering the probability of being harmed the second order uncertainty should also be determined. The second order uncertainty describes the confidence that the value of for instance $P(harm)$ is accurate/correct and will also be referred to as the *confidence level* (see Section 2.1.6). If the second order uncertainty is low the confidence in the accuracy of the information will be accordingly high - hence high second order uncertainty will make $P(harm)$ unreliable. Determining this second order uncertainty is done by the use of trusting beliefs (see Section 2.1.5) which will be described shortly.

Likelihood is in everyday talk normally measured in steps like the following: Rare, Unlikely, Moderate, Likely, and Certain. The degree of consequence is measured: Insignificant, Low, Moderate, Significant, and Catastrophic. In case the likelihood is considered *unlikely* even if the consequence is considered *significant* the risk would not pose a great threat. If the likelihood and degree of consequence is respectively *likely* and *moderate* it might pose a bigger threat although the impact of the threat if realized is less catastrophic than the case not considered a serious threat.

2.1.5 Trusting Beliefs

Measurements of trusting beliefs can be used in estimating how much an entity should be trusted. A belief characterizes the information used when deciding the second order uncertainty of an entity posing some risk, which if realized might harm our assets. In other words beliefs specify the confidence that an entity will do what it is supposed to do, e.g. being trustworthy. Beliefs can for instance be: Honesty, Competence, Benevolence[2], and Predictability as stated in [7]. When specifying a belief one should also state the value of the belief at which point the second order uncertainty is sufficiently low in order to be confident in the given entity.

[2]Likely to behave in a way that is not damaging to us.

2.1.6 Confidence Levels

An entity can pose some risk. Whether to accept this risk should be determined. Determining if a risk should be accepted is done by evaluating the entity posing the risk.

An entity is evaluated by the confidence we have in it, which means how confident we are that it will not harm our assets ($1 - P(harm)$). We need to specify the required confidence in the entity, at which point we are sufficiently confident in the entity, thus the risk could be accepted.

A confidence level describes the confidence required in some entity posing a risk in order to rely on it. The degree of the risk needs to be determined in order to specify this confidence required in the entity posing the risk.

We define the degree of a risk to be the product of the likelihood of the risk being realized and consequence in the given situation (see Table 2.1).

Likelihood/Consequence	**Insignificant**	**Low**	**Moderate**	**Significant**	**Catastrophic**
Rare	Low	Low	Low-Medium	Medium	Medium
Unlikely	Low	Low-Medium	Medium	Medium	Medium-High
Moderate	Low-Medium	Medium	Medium	Medium-High	Medium-High
Likely	Medium	Medium	Medium-High	Medium-High	High
Certain	Medium	Medium-High	Medium-High	High	High

Table 2.1: Degree of risk given likelihood and consequence

The consequence can be measured in many ways; loss of money, resources, or getting private information revealed etc. Therefore the degree of the consequence depends on the person owning the given asset and the specific situation. Loosing for instance $10.000 dollars might be a serious consequence for some but only a minor loss to others. Therefore the degree of consequence is ranging from *insignificant* to *catastrophic*, where oneself decides what the loss of $10.000 dollars corresponds to.

We operate with five confidence levels ranging from *Low* to *High*. Table 2.2 describes the confidence level required in different degrees of risk.

p specifies the probability of an entity to be trustworthy in a given confidence level. For instance an entity with a confidence level of *Medium*, means that the probability for this entity to be trustworthy is 0.7. These probabilities should be stated by the user analyzing the risks and are only indicated here as an example of a distribution.

When for instance having two or more beliefs used to characterize an entity, when combined the beliefs might only require to be in state *Medium* in order to indicate having *High* confidence in the entity. It is up to the user of the system to

Confidence Level	Quantitative	Qualitative
Low	Should only be trusted if the risk is very low	$p \geq 0.3$
Low-Medium	Should only be trusted if the risk is low-medium	$p \geq 0.5$
Medium	May be trusted if the risk is medium	$p \geq 0.7$
Medium-High	May be trusted in medium-high risks	$p \geq 0.85$
High	May be trusted even in high risk situations	$p \geq 0.95$

Table 2.2: Required confidence levels in different degrees of risk.

specify if combined beliefs should be able to result in a higher degree of confidence in the entity, for instance $2 \times \textit{Medium} = \textit{High}$.

2.1.7 Metrics

As beliefs help specify the second order uncertainty of some entity, the beliefs themselves should also be considered for correctness (*their* second order uncertainty). Metrics can improve our confidence in the correctness of some beliefs, resulting in better understanding. If we for instance are not confident in a belief to be correct/accurate, a metric can be used thus getting more information about the given entity. This information can make us more confident in a value of a belief, meaning it can affect the correctness of the belief thus the value of the belief can be affected in both negative and positive way. A metric can be used if a decision is to be made based on some beliefs with inadequate confidence in its accuracy. A metric could for instance be asking someone for an opinion on the given entity, order some kind of analysis on a company thus gaining better/more information on the company, or simply take advantage of some previous experience with the entity. A metric has a cost which is why it should be considered whether or not to use the metric. Metrics themselves should also be trusted, so if its reliability is not known a simple trust management process should be applied to the given metric. The confidence in the given metric should also be stated[4].

2.2 Trust Policy

The result of this trust management process is a trust policy. A trust policy should clearly state the following[4]:

- An overview of all trust metrics
  - The context in which each metric can be used
  - The beliefs in which they can gain confidence

  - The confidence in the metric for the specific context
  - The cost of using the metric

- The confidence levels
  - The confidence levels should be stated, including what probability p the different levels correspond to.

- Trust contexts
  - Context description
  - The risks involved
  - Possible threats
  - Beliefs and confidences required for each context
  - Available metrics for each context

2.3 An Illustrative Example

As an example of how to use a trust management process we will describe a scenario in which the decision whether or not to make an investment is analyzed. This example is based on an example from [4]. In this fictive example only parts of the whole trust management process will be described.

Bob is going to invest some money in a company. The investment will be handled by a second company taking care of investments in shares. In order to decide whether or not to do the investment in this company he uses information from two sources; a friend, Alice and the news bureau Reuters. The entities that should be trusted are therefore Alice, Reuters and the two companies. He identifies his assets to be: his money, his credit card number and his privacy. From these assets he finds the vulnerabilities resulting in the following risks that could lead to harming his assets: *Information Bob uses could be inaccurate*, *Company might go out of business*, and *Credit card details might be revealed*. Having identified these risks they should be evaluated regarding their likelihood and consequence, see Table 2.3.

The likelihood of the risk of getting inaccurate information is not possible to define, whereas the consequence is *significant*, as inaccurate information would cause the investment to be made on wrong basis thus risking to lose money. The likelihood that the company goes out of business is evaluated on bankruptcy tendencies in the given field in which this company operate. The likelihood is deemed

	Likelihood	Consequence
Inaccurate information	Unknown	Significant
Company might go out of business	Moderate	Significant
Credit card details might be revealed	Moderate	Low

Table 2.3: Risk analysis for the investment example specifying likelihood and consequence.

moderate. The consequence is significant and therefore this risk should be considered carefully. Credit card details do get revealed quite often that is why the likelihood is set to moderate. Usually it would have significant or even catastrophic consequences, but since the company bares liability for all but $50 of fraudulent transactions, the consequence is only considered *low*. Next step is to evaluate the required beliefs and confidence levels required. Inaccurate information is considered to have significant consequences, therefore the beliefs, on which we determine whether or not to trust the entity providing the information, should be in state *Medium-High* in order to be confident that the entity is reliable. This is measured by the use of Table 2.1 and Table 2.2 by evaluating the risk to be *Medium-High* which requires a similar degree of confidence in the accuracy. This risk will use the beliefs: honesty and competence in order to determine the reliability of the sources from which he gets information. This should also be made for the two remaining risks; *Company might go out of business* and *Credit card details might be revealed*, but will not be further dealt with in this illustrative example. Eventually metrics available will be defined. As an example can be mentioned: Previous experience (*Medium-High* confidence), Recommendations from other trusted sources (*Medium* confidence), and Certified quality system (*Medium-High* confidence). Each of these metrics has a degree of confidence in which it is trusted to be reliable. At this point the trust policy is to be written but will be skipped in this example.

To summarize, this is a problem regarding trusting other entities in order to make a decision which can have severe consequences for oneself. Every threat should be handled so the risk of it being realized can be reduced. In the next section we will show how Bayesian networks incorporated with trust management can help make such decisions.

2.4 Using Bayesian Networks to Handle Decision Making

The example explained above can, even though the trust management process has been followed, be difficult to handle. Everything has been specified so the problem lies in how to use this information optimally. One way to handle the information

is by generating a Bayesian network[3] for the decision to be made. In a Bayesian network it will be possible to model entities, beliefs, and metrics etc. in a precise and clear manner, making it easier to manage the situation.

2.4.1 From Trust Policy to Bayesian Networks

As the trust management process and the resulting trust policy has been finalized the information gathered should be incorporated in the Bayesian network without compromising any aspects just identified in the trust management process.

Contexts

Each context identified should be illustrated as a Bayesian network since each context is a scenario in which something should be decided. When establishing these networks, a network for each context should be created, each with a decision node and utility node corresponding to the overall decision in the given context/scenario.

Entities & Trusting Beliefs

Each risk identified is posed by some entity. The risk might be either to rely on information granted by the entity or simply to rely on the entity to perform some task. These two types of risks are each modeled differently. Entities that contribute with some information should be evaluated whether or not to be trusted. This is done by examining the already mentioned trusting beliefs. If information is always trusted there is no reason for evaluating the source providing the information. In this case though we do not fully trust the answer/information supplied by the entity. We are only supplied with an answer from the entity and the given trusting beliefs, so by using this information it is possible to give an approximation of what the *true* value is. The true value is the answer the entity would give if the entity was 100% honest and 100% competent and so on.

Figure 2.1 depicts how information from an entity can be evaluated in a Bayesian network. When someone gives a piece of advice or some information, the answer is actually affected by the true value. In the example of Alice giving an advice on the shares, her answer is affected by what the real state of the shares is (the true value) and her competence and honesty. The more confident we are in her, the more credit her answer will be granted. When more than one belief is involved a *confident?* node is used in order ease the creation of the Bayesian network. This node specifies the confidence level in the entity involved.

[3] More information on Bayesian networks and influence diagrams can be seen in [6].

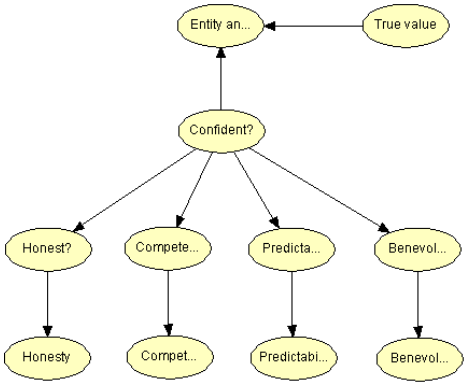

Figure 2.1: Evaluating information from an entity supplying information.

The beliefs will be specified using two nodes, one indicating the value of the given belief (from *Low* to *High*) and one specifying what this value corresponds to in *Yes* and *No* measurements.

Not all risks involve information *from* an entity. Some risks can be evaluated based on facts instead of beliefs. Facts have no second order uncertainty. These should be modeled like an ordinary Bayesian network with a *confident?* node in that entity caused by various facts that has been examined. This information is illustrated in the nodes Fact1 and Fact2 etc. in the model shown in Figure 2.2.

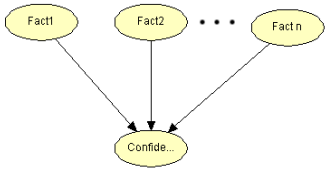

Figure 2.2: Evaluating confidence based on facts.

Confidence Levels

The confidence levels specify the confidence required in different degrees of risk. It states how much for instance *Medium* confidence corresponds to regarding probability in qualitative measures. This knowledge is also required in the Bayesian model, as it is used to determine the probability distribution in the conditional probability tables (CPTs). As an example see Figure 2.1. If the confidence in the entity is *High* then according to Table 2.2, the entity node must have 0.95 chance

of being in the same state as the *True Value* node. If the confidence is medium, the entity node must have 0.7 chance of being in the same state as the *True Value* node, and so on.

Metric

The metrics also have a second order uncertainty. Therefore the confidence in each metric should be evaluated. Since a metric has a cost it should also be possible to evaluate the given situation in order to decide whether to use it or not.

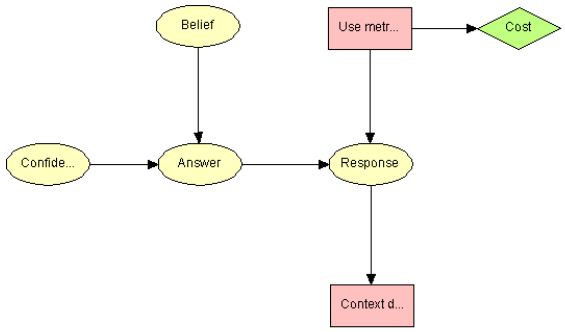

Figure 2.3: Evaluating a metric.

Figure 2.3 shows the structure needed in a Bayesian network in order to model a metric. The *Belief* node is the belief being affected by the metric, whereas the *Confidence* node is the confidence in the given metric which also affects the answer. The *Response* node has three states; Yes, No[4], and Unknown, since in case we decide not to take advantage of the metric the response from it will not be known. Every node in the given Bayesian network that is known at the time the decision should be made, must be connected to the decision node *Use metric?* in order to be able to evaluate the overall situation. The decision node *Context decision* is the main decision node for the given context.

2.4.2 Resulting Bayesian Network

We have modeled part of the example mentioned in Section 2.3 in Figure 2.4 as a Bayesian network. In case the information supplied by Reuters and Alice was 100% reliable the Bayesian network would only consist of the *Share* node and entity nodes.

[4]Yes and No correspond to the states of the answer from the metric.

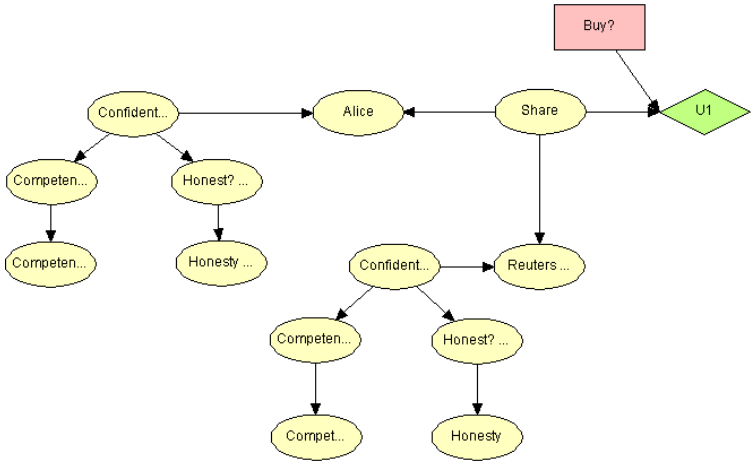

Figure 2.4: Bayesian network for the investment scenario

In order to keep things simple, this model shows the decision of whether or not to buy shares only based on information from Alice and Reuters. Their reliability is indicated by the beliefs: Honesty and Competence. The *Share* node specifies the true probability that the share will increase in value, and should be seen as the probability distribution that we are trying to approximate. This together with the confidence level affects what Alice and Reuters give as advice. In case our confidence level in Alice is very low her answer will not be able to provide very usable information of what the state of *Share* really is. In other words, if the confidence level is *high* the answer from Alice would be a good indication of what the state of *Share* truly is. In case Bob's confidence in both Alice and Reuters are *high* and they give the same answer, he can be very confident in the state of *Share*, thus easing his decision whether or not to invest. To summarize we have the following nodes:

- **Alice:** This node represents the answer received from Alice. It is affected by the two nodes *Confident?* and *Share*. An example of a conditional probability table for this node is depicted in Table 2.4. If Bob is not confident

Confident?	**Yes**		**No**	
Alice / Share	Rise	Fall	Rise	Fall
Buy	1	0	1	1
Don't Buy	0	1	1	1

Table 2.4: CPT for *Alice*

in Alice (which corresponds to having a confidence level of *low*) her answer

will be independent of *Share*. In case Bob is confident in Alice he can rely on her statement and follow her advice, as her answer will be similar to *Share*.

- **Reuters:** This node represents the information received from Reuters. Information from Reuters is like Alice also affected by *Confident?* and *Share*. A conditional probability table similar to Table 2.4 can be used for Reuters.

- **Confident?:** The Confident? nodes each specify our confidence in the given entity to be reliable or not. The Confidence is based upon information regarding the two beliefs: Competence and Honesty.

- **Competence/Competent?:** Competence is indicated by the node *Competent?* with the states *yes* and *no*. The *Confident?* node will receive information stating the probability for *yes* and *no* whether the source is competent from the *Competence* node, ranging from *Low* to *High*, in 3 states. An example of a conditional probability table for the *Competence* node can be seen in Table 2.5.

Competence / Confident?	**Yes**	**No**
Low	0.2	0.8
Medium	0.5	0.5
High	0.8	0.2

Table 2.5: CPT for *Competence*

- **Honesty/Honest?:** Like the *Competence/Competent?* nodes, these nodes specify the honesty of the given entity. A conditional probability table similar to Table 2.5 can be used for the *Honesty* node.

- **Share:** The value of this nodes is what is interesting. It receives evidence from the two entities in order to approximate what the true value is. In case the two entities give wrong advice and whether the entity is confident or not is based on wrong beliefs, the value of this node will be unusable. For this node to be reliable it is therefore important that the second order uncertainty in the entities is as low as possible.

The Bayesian network makes it possible to decide what action to take, given that evidence on competence, honesty and answers from both entities are stated. It is also possible to see how the overall utility is affected by stating different evidence. These possibilities would not be possible by this trust management process alone. The use of Bayesian networks makes complex problems of decision making

more accessible. As soon as the Bayesian network has been established it is easy to model different aspects of the problem and quickly observe tendencies arising.

It is also possible to expand the Bayesian network and be able to for instance evaluate the given situation and decide whether or not an entity should be asked for advice. The goal is to evaluate how much the information from the given entity will affect the current situation. This is done by considering the cost of obtaining that information and the confidence in the entity providing it. A Bayesian network evaluating whether or not to ask Reuters can be seen in Figure 2.5.

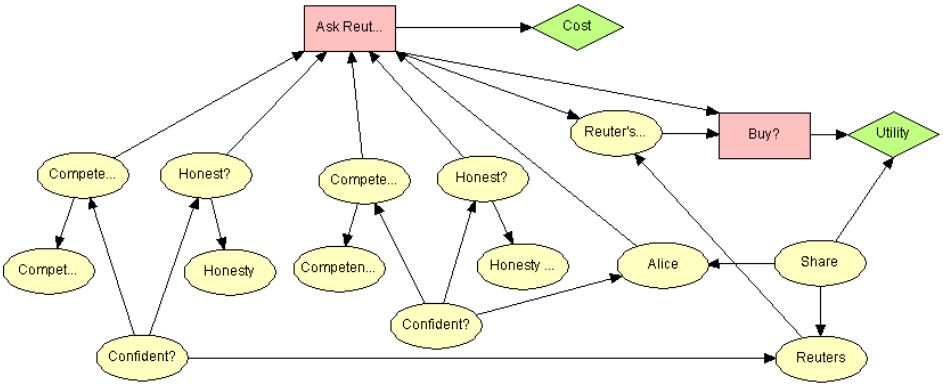

Figure 2.5: Bayesian network for whether or not to ask Reuters.

This model should be seen as an expansion of the previous Bayesian network. In this case only Alice is being asked at first, after which Reuters is evaluated before taking advantage of their contribution, in other words Reuters is here treated in a way similar to a metric. The *Ask Reuters?* decision node is affected by every node which state is known at the time the decision should be made, thus being able to evaluate the overall situation. The utility node *Utility* specifies the cost of asking Reuters for advice, and the *Reuters' Response* node which has the same states as *Reuters* including the state *unknown* in case Reuters is not asked.

Chapter 3

Environment

Before suggesting a solution to the spam problem identified in Section 1.1, it is a good idea to take a brief look at the surrounding environment in which the problem exists. In this case the environment is the Internet and the e-mails themselves.
We will start by looking at the structure of the Internet.

3.1 The Infrastructure of the Internet

The Internet is best thought of as having three levels. At the bottom are local area networks (LANs); for example, campus networks. Usually the local networks are connected to a regional, or mid-level network. The mid-levels connect to one or more backbones. A backbone is an overarching network to which multiple regional networks connect, and which generally does not serve directly any local networks or end-users. The backbones connect to other backbone networks around the world. See Figure 3.1. The connections between the three levels of networks are managed by machines known as routers. It is the responsibility of the routers to route the streams of data from one network to another[5].

There are, however, numerous exceptions to this structure. The separation between the backbone and regional network layers of the current structure is blurring, as more regionals are connected directly to each other through network access points (NAPs), and traffic passes through a chain of regionals without any backbone transport.

3.2 E-mails

Just as there are certain rules to obey if a letter is to reach its addressee (size, weight, stamp, specification of address, etc.), certain demands have to be met with

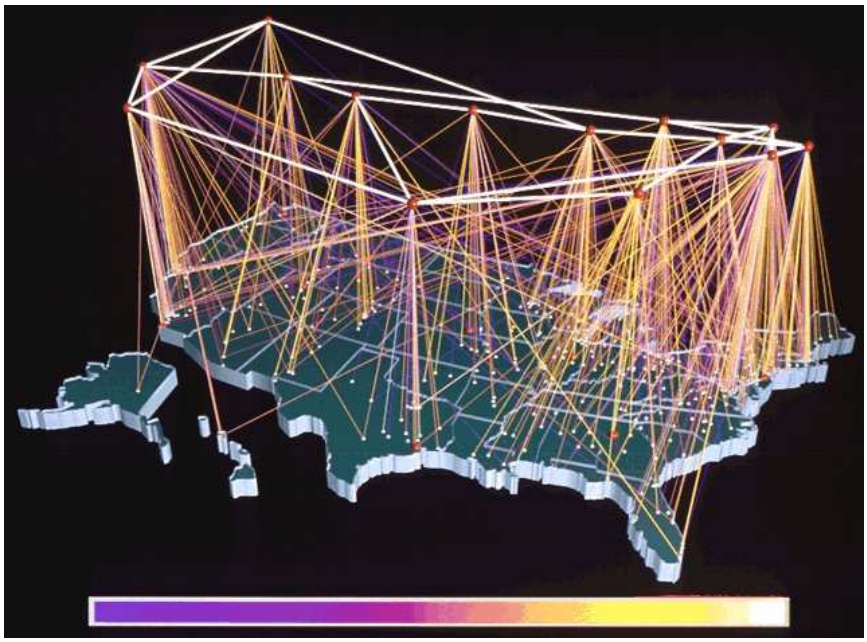

Figure 3.1: Internet structure in North America[9]

electronic mail.

Analogue to letter mail an e-mail consists of:

- *Envelope:* The envelope is needed by the e-mail system (more exactly the Mail Transport Agent (MTA)) for e-mail delivery. It contains the address(es) of the recipient(s) and hints about the relay stations the e-mail passed through.
- *Header:* The header contains information to the receiver of the e-mail. This information is normally things like: who sent the message, at what time was the message sent, the subject of the message and so on.
- *Content (body):* The content or body of an e-mail is the message itself.

When an e-mail is sent from a computer it is sent using SMTP (Simple Mail Transfer Protocol) and TCP (Transmission Control Protocol).

As mentioned in the introduction, some people are sending a lot of advertising e-mails to other people who do not want them. Thus spam filters were invented to sort out the desirable e-mails from the spam. We will now take a brief look at the spam filters themselves.

3.3 Types of Spam Filters

This section will examine how spam filters work and which kinds of filters are available today. The two main categories are server-side and client-side spam filters.

Server-side spam filtering catches spam e-mails before they reach their destination. A serious problem regarding spam filters is that often false positives[1] get caught as spam. This scenario is considered worse than actually receiving the spam, since you might loose important e-mails that will be non-recoverable. Therefore server-side spam filters are often configured to be weak and only catch e-mails that contain many elements categorizing it as spam.

Client-side spam filtering has the same problem as server-side filtering, but the consequence of a false positive is not that serious since on the client side you still have access to those e-mails identified as spam no matter if they are spam or wrongly identified e-mails. Not having this concern makes it possible for the filter to be configured more strict, by which is meant that less elements categorizing as spam needs to be present in order to identify an e-mail as spam.

As can be seen, server-side filtering has some flaws that prohibits it from being superior in spam identifying. This does not mean that server-side filtering should not be used, although it is bad in identifying spam, it still removes many spam e-mails resulting in less bandwidth getting wasted. Therefore server-side filtering should be used in conjunction with client-side filtering.

Each of these filters uses different methods[1] to identify whether an e-mail is spam or not:

- *Header filter:* This sort of filtering method examines the header and envelope of the e-mail to see if something in the relay chain has been forged so it cannot be traced.

- *Language filter:* This kind of filter simply denies e-mails written in languages other than what the user specifies. This will only remove spam e-mails written in foreign languages and is therefore not very helpful in itself. It can also be difficult to say that you will never receive e-mails in another language that is not important. It should be noticed that the majority of spam is in English.

- *Content Filter:* Filtering after content requires that the whole e-mail gets scanned and every word checked. Every word is then assigned some score. Words that receive high scores are often associated with spam e-mails, thus

[1] A false positive is an e-mail wrongly classified as spam.

e-mails resulting with a high score will be identified as spam. The problem is deciding what this threshold should be. Too high and you end up trashing your non-spam e-mails too (false positives), but too low a threshold will result in bad filtering and you will still receive spam (false negatives).

- *Permission filter:* Permission filters will only accept e-mails from people that you give permission, for instance through a web site on which they will need to register and be accepted by you for them to send you e-mails. This results in a perfect filter but will not be practically reasonable because of the amount of work involved in accepting/rejecting the many requests.

Methods and filters just described is how the most widespread spam filters work today. One of these is SpamAssassin[2] which utilizes many methods in order to make a good classification of e-mails. It works quite efficiently but is not perfect at all. It is possible to loose important e-mails trashed as spam, and quite a lot of spam will still find its way to your mailbox. And why is that? Because SpamAssassin uses deterministic rules to classify spam. This means that those people sending out spam just need to study these rules to avoid those characteristics in their "advertising".

Lately Graham[3] has proposed a spam filtering strategy using a statistical approach where instead of giving scores to words it assigns a probability of whether this word is associated with spam. This probability table is generated from a large collection of e-mails, both spam and non-spam. Every e-mail is categorized to be either spam or non-spam thus affecting the probability assigned to the given word. Whenever a user receives spam (false negative) he can mark it as spam and the filter will scan that e-mail and adjust its probability table accordingly using Bayes' rule thus being able to capture future spam. By doing this we do not suffer from static scoring rules, like in normal filtering, since it adapts to new kinds of spam on the fly, which results in less false positives and false negatives. This makes it more difficult for the spammers to generate spam that will not be caught. Another nice feature is that it will work as a personal spam filter optimized to the kind of e-mails *you* receive so that the occurrence of false positives is much more unlikely.

To summarize, the statistical approach reduces the number of false positives greatly, and helps getting rid of the static scoring rules, although the scoring threshold still needs to be decided. Besides that it is much more efficient because of its adaptability and will hopefully help reducing the phenomenon "spam". The only aspect that is not taken care of with the statistical approach is the mentioned waste of bandwidth by using client-side spam filters. It does not seem to be practically as a server-side filter, since it would lose one of its main advantages, the ability to adapt to the individual user's preferences.

As specified in Section 1.1, we will develop a server-side spam filtering system that reduce the resources required by the client-side filters while at the same time regain the possibility of individual user preferences.

Chapter 4

First Stage

The goal is to develop a spam filtering system making it possible to remove spam e-mails as early as possible thus saving resources. This system will, as mentioned in Section 1.1, be called TrustOne. We will start our design of TrustOne by creating the simplest design we believe has some chance of success. We will refer to this simple design as the *basic design*. Considerations such as the amount of resources needed to run TrustOne, or the bandwidth needed, will not be examined in this stage. Such considerations will be made later in more advanced and realistic versions of the design. This basic design will not contain any of the trust aspects that we believe is necessary for the design to be effective, trust will be introduced in later revisions of the design, see Chapter 6. We spend time on this first design only to get the basic elements of the system in place.

4.1 The Basic Design

The main difference between TrustOne, and the currently used spam filters, is the use of communication regarding spam between computers. In other words, outsourcing of the spam filtering process only works if the participating computers can communicate and in some way coordinate their efforts. In this our basic design the only form of communication possible is a response when a spam e-mail is received. The response is sent to the computer who handled the e-mail last.

As an example (see Figure 4.1): Machine F wants to send an e-mail to machine B. F thus delivers the e-mail to either machine D or E. Assume that the e-mail is delivered to E. E then forwards the e-mail to its destination, machine B. B now runs the e-mail through its spam filter. If B classifies the e-mail as spam, it reports this classification to the machine who handled the e-mail last, in this case machine E.

This classification-report will tell E that B received an e-mail from E, and that

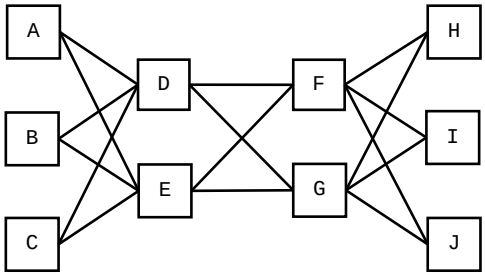

Figure 4.1: Model of interconnected mail servers

B considered that e-mail as spam.

If *E* receives enough reports from other machines that they receive spam delivered by *E*, *E* will then activate its own spam filter in order to examine where it receives all these spam e-mails from. Now, every time *E* receives an e-mail that *E* considers spam, it deletes the e-mail and sends a classification-report to the machine who delivered it, just as *B* sent a classification-report to *E*. Since *E* is now running a spam filter, there is no longer a need for machine *B* to run its spam filter on e-mail delivered from *E* as these e-mails have already been through a spam filter. Thus machine *B* can save the resources normally spent on filtering e-mails from *E*. Also the bandwidth of the connection between *B* and *E* will no longer be used to deliver spam.

This cycle then repeats. If *F* receives classification-reports from *E* and possibly other machines, *F* starts its spam filter, thus removing the need for *E* to run its spam filter on e-mails from *F*, and so on.

After several of these cycles it will be possible to only run spam filters on e-mails from those machines that actually send spam, and skip the filtering process on e-mails from machines that are not reported to be spamming. This would lower machine overhead and still keep spam away. The problem of wasted bandwidth is improved but not solved using this basic design. Some spam e-mails will be stopped resulting in less bandwidth consumption, but all the classification-reports that are sent back and forth will require some bandwidth and will also give more work for the machines that are to generate or receive/compute these reports.

For this design to work, we have to make some assumptions:

- *Routers:* The Internet routers must be able to run spam filters and to send/receive and process the classification-reports.
- *No false positives:* By allowing other machines to filter e-mails for spam, control is lost. Since only classification-reports are communicated between the machines, the person for whom an e-mail is intended has no way of

checking whether he agrees with a machine's classification of spam e-mails. If a machine classifies an e-mail as spam and removes it, the intended receiver will never know about it, and he can therefore not complain if the e-mail was in fact *not* spam. Thus the spam filters in the basic design, must *not* cause false positive classifications of e-mails.

- *Same classification:* By moving the spam filter away from the receiver of the e-mails, we lose the ability to allow individual configurations of the spam filters. In the example above, if machine E starts its spam filter, all e-mails passing through E will be filtered using the same configuration, that is, E's configuration. Since E knows nothing about how other machines classify spam, all machines need to have the same classification or we risk having false positives.

- *No false classification reports:* No machine may send false classification-reports. If a machine for some reason starts to send false reports to other machines, claiming that it receives a lot of spam from them, the other machines will start their spam filters, thus using unnecessary resources.

The *Routers* assumption is very hard to avoid, since this functionality lies at the very core of the TrustOne idea. Changing the responsibilities of the routers might at first seem as a very serious assumption that makes the design useless in real life. When the current responsibilities of routers are examined however, we find that they already have other functionality than just transferring packets from one network to another. Many routers are running software that makes the router able to better do its job. They analyze the traffic flow of the packets and do complicated calculations in order to improve their performance. Thus the assumption that the routers should be able to run spam filters is not that great a change from their present responsibilities.

The other assumptions however can be reduced or removed by improving the basic design. The improvements emerge from more advanced communication between the machines and the use of Bayesian Networks to help the routers in their decisions, see Chapter 5.

Chapter 5

Second Stage

In this chapter we will extend the basic design described in Chapter 4. We will reduce the resources required by the basic design and at the same time make the design more suited for the real world by removing or reducing the assumptions required by the basic design. The extended design will accomplish this by introducing a more complex communication protocol between the machines and also by using Bayesian Networks in the decision making processes.

5.1 Extended Design

In the basic design, every time an e-mail is classified as spam a classification-report is sent to the last machine who handled the e-mail. All these classification-reports use some of the resources that we are trying to save.

The extended design thus handles the communication between the machines a bit different. When for instance machine *B* receives an e-mail delivered by machine *E*, it runs the e-mail through its spam filter. Instead of sending a classification-report for each received spam e-mail, *B* simply stores the spam e-mail together with the other spam e-mails that *B* has received from any machine, not just *E*. If the e-mail was not a spam e-mail, it is stored together with the other non-spam e-mails that *B* has received from any machine. In this way *B* will gather a collection of spam and non-spam e-mails based on *B*'s classification. *B* also keeps track of how many e-mails it has received from *E*[1] and how many of these were spam. If *B* decides that too many spam e-mails are coming from *E* it sends a message to *E* requesting *E* to run a spam filter on all the e-mails that *E* is delivering to *B*. Together with this request, *B* sends several pieces of information:

[1] As well as from any other machine from which *B* receives e-mails.

- The total number of e-mails *B* receives from *E*.
- How many of these e-mails are considered spam.
- *B*'s collection of e-mails classified as spam[2].
- *B*'s collection of non-spam e-mails[3].
- *Optional:* A suggestion of what type of spam filter and filter configuration *E* should use. The suggestion is optional, since not all machines will be able to give such a suggestion in advance.

Machine *E* must now decide whether it wants to accept or deny *B*'s request. To make this decision *E* uses the *Provide filtering* Bayesian network described in Section 5.2.1.

E has three options:

- *Reject: E* can of course say that it does not want to run a spam filter on *B*'s behalf. In this case, nothing more happens. We cannot force someone else to run a spam filter on our behalf.
- *Accept suggestion:* If *B* makes a suggestion to the type and configuration of the spam filter, *E* can just start running the spam filter following that suggestion. We can safely assume that the *B* will be satisfied with the spam filter, since we are following *B*'s own suggestion.
- *No/reject suggestion:* If *B* does not make a suggestion to the type and configuration of the spam filter, or if *E* rejects the suggestion, *E* can start a filter of a type and configuration of its own choosing. In this case, *E* has to make sure that *B* approves of the spam filter chosen.

The *No/reject suggestion* option is a bit more complex than *E*'s other two options, and will be handled in detail in Section 5.1.1.

5.1.1 No/Reject Suggestion

E has decided that it will not use the, by *B*, suggested type and configuration of spam filter. *E* must now make a counter-suggestion to *B*.

The main problem for *E* is now to find a type and configuration of a spam filter that *B* will find acceptable. To this end, *E* uses the collections of spam and

[2]If *B*'s collection is very large, *B* only needs to send a representative subset of the spam e-mails.

[3]If *B*'s collection is very large, *B* only needs to send a representative subset of the non-spam e-mails.

non-spam e-mails supplied by *B*. These collections represent the type of e-mails normally received by *B* and thus if *E* can find a type and configuration of spam filter that can correctly classify the supplied collections, *E* can increase the chance that *B* will be satisfied by the choice of the spam filter.

If *E* cannot find a type and configuration of spam filter that it believes *B* will accept, it informs *B* that it cannot fulfill *B*'s request to run a spam filter on *B*'s behalf. On the other hand, *E* can send a message to *B* containing:

- The type and configuration of the proposed spam filter.
- How well the proposed spam filter classified the collections supplied by *B*.
- Various other information.

Various other information could be almost anything. A couple of examples could be:

- *Availability of the spam filter:* Can *E* for instance guarantee that all e-mails it delivers to *B* will pass through the spam filter, or will *E* turn off the spam filter for periods of time, when other higher priority tasks requires *E*'s resources.
- *Storage of removed spam e-mails:* Will *E* store the e-mails it removes as spam. If so, for how long will they be stored. Such a feature could for instance be useful if *B* suspects that *E* has made some false positive classifications of its e-mail.

Other types of information could also be useful but we will not try to predict them all here.

It is now up to *B* to decide whether to accept *E*'s offer. First *B* checks if it can accept the information supplied under *Various other information*. If not, *B* informs its decision to *E*, and *E* will not be running a spam filter on the behalf of *B*. If *B* finds the information acceptable, it must decide whether it finds the classifications made by *E* sufficiently correct. The *Allow Filtering* Bayesian network described in Section 5.2.2, will help *B* in its decision. *B* then informs *E* of its decision. If *B* is satisfied, *E* starts running the spam filter on *B*'s behalf. If not, then nothing more happens.

If *E* receives requests to run a spam filter, it indicates that at least some spam e-mails are passing through *E*. *E* can then decide if it wants to examine from which machines it receives this spam, and whether to request these machines to run spam filters on *E*'s behalf. The entire decision cycle then repeats.

The effect of this extended design will be the same as for the simple design, to filter out the spam as close to the source of the spam as possible, thus saving resources. The extended design has some advantages though compared to the basic design:

- *Individual filtering:* Since a machine like *B* must request *E* to run a spam filter, *E* only needs to run the spam filter for those machines who request it. This reduces the workload on *E*.
- *Individual spam filter configuration:* Since *B* suggests the type and configuration of the spam filter, *B* still retains some of the influence on the spam filter normally lost when moving the filter away from *B*. Even if *E* rejects *B*'s suggestion, *B* still has to accept a spam filter that *E* will be using on e-mails to *B*.
- *Less messages required:* The extended design requires fewer messages than the basic design between the machines in order for the design to work. Messages are only sent when one machine is considering allowing another to filter its e-mails.
- *Distributed configuration:* Since machines that ask a machine to run a spam filter on their behalf also send the preferred configuration of the spam filter, the receiving machine can use these configurations to get an idea of what configuration it itself can ask yet another machine to use. As an example (see Figure 4.1): *E* runs spam filters for *B* and *C*. *E* has not heard from *A* and it can thus assume that *A* is satisfied with the situation as it is now. *E* discovers that a lot of the spam is delivered by machine *F* and will therefore request that *F* starts a spam filter on *E*'s behalf. *E* can now use the configurations and requirements of *B* and *C*, together with its own requirements, to determine how *E* would like to have F run the spam filter.

The extended design also has a few assumptions that must be obeyed for it to work properly.

- *Routers:* The Internet routers must be able to run spam filters and to handle the communication protocol between the machines.
- *Must request filtering:* If a machine is unhappy with the amount of spam it receives, it must request the machines who deliver the spam to run a spam filter. If for instance machine *B* is unhappy with the amount of spam it receives from *E* but it does not request *E* to run a spam filter. *E* can then only assume that *B* is happy with the situation, and thus *E* might never know that it delivers spam to others.

- *A machine must not lie:* The machines must not lie to one another as this might result in loss of performance and increase the risk of filtering false positives e-mail. The Bayesian networks in their current form will simply not be able to help the machines in their decision making if the evidence entered into the networks is false.

As in the basic design, the *Routers* assumption is very hard to avoid. The *A machine must not lie* assumption is undesirable and can be removed/reduced by introducing trust into the design. This introduction of trust is the main focus of the next and final stage of the design, see Chapters 6 and 7.

5.2 Description of Bayesian Networks

As mentioned some decisions are taken using Bayesian Networks. The decisions needed in the TrustOne system are the following:

- *Provide Filtering:* This decision is taken by the machine that is asked to do spam filtering on another machine's behalf. The Bayesian network helps the machine to decide whether or not to provide the filtering by examining several aspects.
- *Allow Filtering:* This decision is taken by the machine who wants another machine to run a spam filter in its behalf. The Bayesian network helps the machine decide whether the spam filter the other machine is willing to run is sufficiently efficient.

5.2.1 Provide Filtering

The Bayesian network in Figure 5.1 for the *Provide Filtering*-decision is used when a machine, for instance machine *E*, is asked to do filtering for e.g. machine *B*. Machine *E* has to decide whether it will accept or reject *B*'s request. *E* needs to consider the following aspects:

- **Resources required:** *E* needs to consider the amount of resources required if *E* is to run a spam filter on *B*'s behalf. The resources considered is CPU usage and IO activity, as these two types of resources are often in short supply.
- **Efficiency of spam filter:** *E* needs to consider whether the spam filter it will use, is sufficiently efficient in removing spam. If the spam filter is not sufficiently efficient, *E* cannot justify the expenditure of resources.

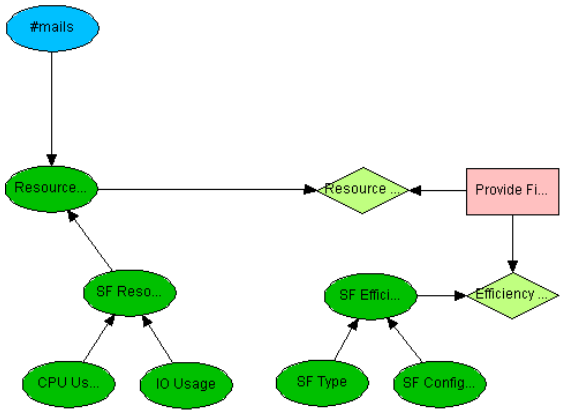

Figure 5.1: Bayesian network for *Provide Filtering*. SF is short for Spam Filter.

We will now take a closer look at the structure of the Bayesian network *Provide Filtering*. In the description of the nodes of the network, *SF* is short for Spam Filter. Some of the names of the nodes are longer that can be displayed in Figure 5.1, and thus only the first part of the name is shown in some nodes.

Resources Required

The part of the *Provide Filtering* network used in deciding the resources required by one machine to run a spam filter consists of the following nodes:

- **Resources Required:** This node represents the total amount of resources needed to run a spam filter on behalf of another machine. The node has 3 states: Low, Medium, High, and is dependent on two other nodes: *#mails* and *SF Resource Use*.

- **#mails:** This node represents the number of e-mails that will need to pass through the spam filter. It has 3 states: Low, Medium, and High. The number of e-mails is supplied in the request a machine receives when asked to run a spam filter on behalf of someone else.

- **SF Resource Use:** This node represents the amount of resources needed to send a single e-mail through a spam filter. The node has 3 states: Low, Medium, and High. The node is dependent on the nodes: *CPU Usage* and *IO Usage* since these two nodes represents the two main areas where spam filters use resources.

- **CPU Usage:** This node represents the amount of CPU resources required to send a single e-mail through the spam filter. The node has 3 states: Low, Medium, and High.
- **IO Usage:** This node represents the amount of IO resources[4] required to send a single e-mail through the spam filter. The node has 3 states: Low, Medium and High.

Spam Filter Efficiency

This part of the *Provide Filtering* network considers whether the chosen spam filter is sufficiently efficient. It consists of the following nodes:

- **SF Efficiency:** This node represents the efficiency of the selected spam filter. This is not the resource efficiency of the spam filter, but how well the spam filter correctly classifies e-mails. The node has 3 states: level1, level2, and level3. Since the efficiency of a spam filter is dependent on the type of spam filter, and how this spam filter is configured, this node is dependent on the two nodes: *SF Type* and *SF Configuration*.
- **SF Type:** This node represents the type of spam filter used. Currently the node has 5 states: Type1, Type2, Type3, Type4, and Type5. All spam filters used with TrustOne must fit into one and only one of these 5 types and all machines must agree on this.
- **SF Configuration:** This node represents the configuration of the spam filter used. Currently the node has 5 states: Conf1, Conf2, Conf3, Conf4, and Conf5. The configuration of all spam filters used with TrustOne must fit into one and only one of these 5 states and all machines must agree on this.

The Utility and Decision Nodes

To be able to make a decision whether a machine should run a spam filter on behalf of another machine, the *Provide Filtering* network is supplied with two Utility nodes and a single decision node.

- **Provide Filtering:** This decision node represents the decision needed, when a machine contemplates running a spam filter on behalf on another machine. The decision has two states: Yes, No and is dependent on the two utility nodes: *Resource Utility* and *Efficiency Utility*.

[4]Hard disk usage. Since hard disks are slow compared to the other parts of a computer, heavy hard disk usage can result in a bottleneck seriously reducing the performance of the computer.

- **Resource Utility:** This utility node represents the cost or gain, based on the states of the decision node *Provide Filtering* and the chance node *Resources Required*.

- **Efficiency Utility:** This utility node represents the cost or gain, based on the decision node *Provide Filtering* and the chance node *SF Efficiency*.

5.2.2 Allow Filtering

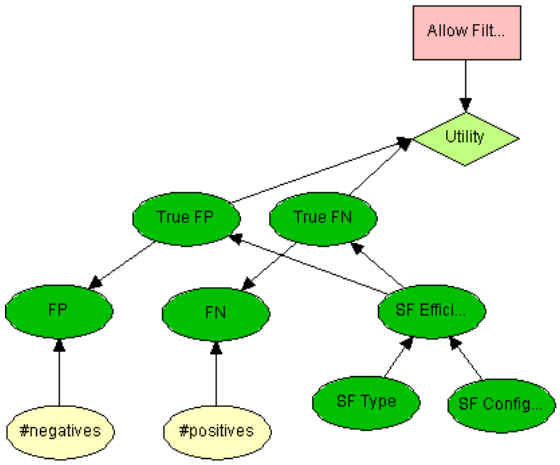

Figure 5.2: Bayesian network for *Allow Filtering*. SF is short for Spam Filter, FP is short for False Positives, and FN is short for False Negatives.

The Bayesian network in Figure 5.2 for the *Allow filtering*-decision is used by a machine when, for instance machine *B*, must decide whether another machine, for instance, machine *E*, is good enough at classifying e-mails on *B*'s behalf. For *E* to be 'good enough', *B* must consider whether the number of falsely positive and falsely negative classified e-mails, made by *E*, is within acceptable limits. *B* needs to find these two numbers as they are at the very core of *B*'s decision.

B has two different pieces of information that can help find these numbers.

- *Machine E's results:* Machine *E* has classified all the e-mails in the two e-mail collections provided by *B*. *E* informs *B* of the result of this classification.

- *Spam Filter Efficiency:* Machine *E* also informs *B* of the spam filter type and configuration *E* used when it classified the e-mails provided by *B*.

We will now take a closer look at the structure of the Bayesian network *Allow filtering*. In the description of the nodes of the network, *SF* is short for Spam Filter, *FP* is short for False Positives, and *FN* is short for False Negatives. The names of some of the nodes are longer that can be displayed in Figure 5.2, and thus only the first part of the name is shown in some nodes. Some nodes represent the amount of something. For instance the *True FP* node represents the amount of false positive e-mails. These nodes have states like: *Many* and *Few*. Just how many 'Many' e-mails is, is dependent on the user of TrustOne.

- **True FP:** This node represents the true number of false positives that *E* will make, if it runs the spam filter on *B*'s behalf. The node has four states: Many, Some, Few and Very Few. *True FP* is dependent on the *SF Efficiency* node, since the true number of false positive classifications made, is dependent on how well the spam filter used by *E* classifies e-mails.
- **True FN:** This node represents the true number of false negative classifications that *E* will make, if it runs the spam filter on *B*'s behalf. The node has 4 states: Many, Some, Few, and Very Few. *True FN* is dependent on the *SF Efficiency* node, since the true number of false negative classifications made, is dependent on how well the spam filter used by *E* classifies e-mails.

As mentioned earlier, at the time *B* makes its decision, it has two sources of information available when trying to predict states of the *True FP* and *True FN* nodes.

Machine E's Results

The part of the *Allow Filtering* network, used to find the true number of false positives and false negatives based on the results of another machine, consists of the following nodes:

- **#positives:** This node represents the size of the collection of spam supplied to *E* by *B*. It has 3 states: Few, Some, and Many.
- **#negatives:** This node represents the size of the collection of non-spam supplied to *E* by *B*. It has 3 states: Few, Some, and Many.
- **FP:** This node represents the number of false positive classifications experienced by *E*, when running the indicated spam filter on *B*'s behalf. The node has 4 states: Many, Some, Few, Very Few. Since the precision of *E*'s result is dependent on the true number of false positives and the size of the collection on which this result was found, the *FP* node is dependent on the *True FP* and *#negatives* nodes.

- **FN:** This node represents the number of false negative classifications experienced by *E*, when running the indicated spam filter on for instance *B*'s behalf. The node has 4 states: Many, Some, Few, Very Few. Since the precision of *E*'s result is dependent on the true number of false negatives and the size of the collection on which this result was found, the *FN* node is dependent on the *True FN* and *#positives* nodes.

Spam Filter Efficiency

This part of the *Allow Filtering* network is used to find the true number of false positive and false negatives based on the type and configuration of the spam filter used in the classification.

This part of the network is identical to the same part of the *Provide Filtering* network, and thus will not be repeated here.

The Utility and Decision Nodes

To be able to make a decision whether a machine is sufficiently good at running a spam filter on our behalf, the *Allow Filtering* network is supplied with one utility node and a decision node.

- **Allow Filter:** This decision node represents the decision needed, when a machine contemplates whether another machine is good enough at classifying e-mails on its behalf. The decision has two states: Yes, No and is dependent on the utility node: *Utility*.

- **Utility:** This utility node represents the cost or gain, based on the states of the decision node *Allow Filter* and the chance nodes *True FP* and *True FN*.

As seen throughout these Bayesian networks we assume that information from other machines is trustworthy which might not always be the case. The need for some trust management is therefore required. Incorporating trust management requires analysis of what sources need to be trusted. In order to determine if something can be trusted, the risk involved within the given situation needs to be specified. In the following chapter trust management will be examined.

Chapter 6

The Trust Management Process

Throughout the previous stages of the design it was assumed that information received from every source could be trusted. Since that might not always be true we need to be able to incorporate trust management into our Bayesian networks making it possible to take risks into consideration. We will analyze the scenarios concerning our situation which will result in a trust policy[4] describing what and how much entities should be trusted in order to avoid risks being realized. This trust policy will help specifying every detail of the entities that should be trusted so incorporating this knowledge into the Bayesian networks will be easy and precise.

The trust policy will contain the following three aspects: trust metrics, confidence levels, and trust contexts. Further details about these aspects are explained in Chapter 2.

6.1 Available Trust Metrics

Trust metrics are used in order to improve confidence in some beliefs, and can be used if something should be trusted but you need to be more confident whether or not it can be trusted. In our situation we have the trust metric: *Previous experience*. Asking another for advice could also be modeled as a metric if the question concerns gaining confidence in the beliefs of an entity. As TrustOne is not going to ask machines about the beliefs of other entities, the task of asking other is not a metric as defined in Section 2.1.7.

6.1.1 Previous Experience

Previous experience can improve your confidence in beliefs such as: honesty, competence, benevolence, and predictability. It can be used in the two contexts: *Pro-*

vide filtering and *Allow filtering*. The metric has some confidence associated with it, describing how confident we are that information from it is precise. In these two contexts the confidence in the previous experience is considered as *Medium-High*. For instance, a metric with *low* confidence would not seriously improve the overall confidence in the given beliefs that are affected. The cost of using this metric is considered *low*, since the effort required, in order to take advantage of our own previous experience/knowledge, has practically no resource usage or other expenses. To summarize, the *previous experience* metric has the following characteristics:

- Contexts in which it can be used: Provide Filtering, Allow Filtering
- Beliefs it can affect: Honesty, Competence, Benevolence, and Predictability
- Confidence in this metric for each context that it can be used: Provide Filtering: *Medium-High*, Allow Filtering: *Medium-High*
- Cost of using this metric: *Low*

6.2 Confidence Levels

Confidence levels are used to indicate the required confidence in some entity given the degree of risk. In Table 6.1 the relationship between the confidence levels, degree of risk and the probability p, is shown.

Confidence Level	Quantitative	Qualitative
Low	Should only be trusted if the risk is very low	$p \geq 0.3$
Low-Medium	Should only be trusted if the risk is low-medium	$p \geq 0.5$
Medium	May be trusted if the risk is medium	$p \geq 0.7$
Medium-High	May be trusted in medium-high risks	$p \geq 0.85$
High	May be trusted even in high risk situations	$p \geq 0.95$

Table 6.1: Confidence levels

6.3 Trust Contexts

Having examined the trust metrics and confidence levels we will now describe the different trusting contexts. The trust contexts are all scenarios, in this case: *Provide Filtering* and *Allow Filtering*. Now every scenario will be described in more detail in order to identify entities, associated risks, required trusting beliefs, and confidences.

6.3.1 Provide Filtering

In this scenario the asset to protect is the resources, since waste of resources is a threat that if realized would remove the advantages of TrustOne whose main objective is to minimize resource usage. The agents posing this threat are spammers and hackers[1]. Trusting entities tell us what in the given situation needs to be trusted. In this scenario we have the following trusting entities: The requester[2] and our own spam filter. The requester might send false requests or might equip us with other inaccurate information. Our own spam filter is considered as a trusting entity since having a badly configured spam filter might also pose a threat. By analyzing the vulnerability of the system, taking the asset to protect and the trusting entities into consideration, the following risks are identified:

- False Requests
- Inefficient Spam filter
- Inaccurate information

These risks will be analyzed regarding their likelihood and consequence (impact) which can be seen in Table 6.2.

Risk	**Likelihood**	**Consequence**
False requests	Moderate	Significant
Inefficient Spam filter	Unknown	Significant
Inaccurate information	Unknown	Significant

Table 6.2: Risk analysis for *Provide filtering* specifying likelihood and consequence.

The reason for analyzing the identified risks is to give an estimate on the degree of the risk, thus being able to find the required confidence in the given entity. The confidence required is reflected in the values of the beliefs. From this table (6.2) it can be seen that the likelihood for *False requests* is considered moderate with a consequence degree at significant. This means, since it is quite likely to happen and still has a consequence degree considered as significant, we need to be confident in the entity posing this risk in order to avoid the threat being realized. Regarding *Inefficient spam filter* and *Inaccurate information* the likelihood is set to unknown since it depends on specific information regarding the given spam filter and what

[1]The threat posed by hackers will not be examined further as protection against hackers is not our main focus.

[2]The machine sending a request to do spam filtering will be called the *requester*. The machine receiving this request will be called the *receiver*.

and from who information is gathered. Their consequences are considered significant.

As just mentioned, this risk analysis can be used to specify the required confidence in each entity in order to rely on it. Each risk has an associated entity posing the risk which also has some beliefs in order to characterize it. The risk of getting false requests is associated with the beliefs: Honesty and Competence. By considering the risk analysis specifying that the consequence was significant and the likelihood for it to happen was moderate, the degree of the risk is evaluated to *Medium-High* according to Table 2.1, hence a confidence level at *Medium-High* is required. The risk; Inefficient spam filter is caused by the entity; *Own spam filter* and therefore has the facts: Type and Configuration, and the risk; Inaccurate information has the associated beliefs: Honesty and Competence. Since the likelihoods for these two risks are unknown, it is not possible to give an *exact* value of what the confidence in their entities should be. In this case though we will give an estimate on the confidence required in their entities to be respectively *Medium* and *Medium-High*, only based on the degree of consequence. The reason that a higher confidence level in the entity for Inaccurate information is required, is because we assume that the threat of receiving bogus information is more likely than having a misconfigured spam filter.

Since we are not always confident about what the right value should be for the beliefs we can take advantage of the already mentioned metric. In this case it is possible to take advantage of *Previous experience*. As mentioned the confidence in this metric is: *Medium-High*.

6.3.2 Allow Filtering

In the Allow filtering scenario we have several assets to protect. These are e-mails, privacy and resources. The reason why e-mail is an asset is because e-mails (false positives) might be lost by the filtering process, or spam (false negatives) might be received due to an inefficient spam filter. These are both unacceptable, although false negatives (spam) are more acceptable than losing e-mails (false positives). By the asset privacy, we mean that private information in e-mails might risk being revealed. As with the *Provide filtering* scenario, resources is also an important asset. Waste of resources might be caused by inaccurate information from both the receiver and information from other machines being asked for advice. The agents posing these threats are spammers and hackers (denial of service, thus wasting resources). To summarize we have the following risks:

- Disclose private information
- Inefficient spam filter

- Inaccurate information (receiver)
- Inaccurate Information (other machine)

The following entities should therefore be trusted: The machine (receiver) that is asked to do filtering, and other machines that might be asked for advice.

The risk analysis for the *Allow filtering* scenario, showing likelihood and consequences for the risks identified, can be seen in Table 6.3.

Risk	**Likelihood**	**Consequence**
Disclose private information	Moderate	Significant
Inefficient Spam filter	Unknown	Significant
Inaccurate information (receiver)	Unknown	Significant
Inaccurate information (other machine)	Unknown	Significant

Table 6.3: Risk analysis for *Allow filtering* specifying likelihood and consequence.

Having specified their likelihood and degree of consequence we need to specify which beliefs are used to determine each risk, the required confidence level of the entities, and thus the required values of the beliefs. For instance the *Disclose private* information risk is posed by the entity *receiver* which has the belief: Honesty, with a confidence level set to *Medium-High* required in the entity because the risk is considered to be *Medium-High* according to Table 2.1. The spam filter efficiency is based on the facts Type and Configuration in which entity we need a confidence level at *Medium*. *Inaccurate information* from the receiver is based on the belief: Honesty, and we estimate the required confidence in the entity to be *Medium* in order to trust the entity posing this risk, in this case the receiver. *Inaccurate information* from other machines is posed by the entity *other machine* which has two beliefs: Honesty and Competence because, in order to evaluate information/knowledge from other machines, we need to be both able to rely on the machine and know how competent solving the given task it is. The confidence level needed in order to trust this entity is estimated to be *Medium*. As with the *Provide Filtering* scenario, a metric can also be used in this scenario if more confidence in the beliefs is required. It is as mentioned: Previous experience (*Medium-High*).

This trust management process of these scenarios has helped locating the entities who need to be trusted in order to avoid the posed risks and the required values of the trusting beliefs. The following chapter will specify the new Bayesian networks taking this knowledge into consideration.

Chapter 7

Third Stage: Incorporating Trust Management

When machines communicate, we cannot be sure that the information communicated is always 100% correct. A machine might outright lie, and thus have questionable honesty or it may simply not have the correct information for some reason thus having questionable competence. In this third stage we incorporate the knowledge about risks, entities and trusting beliefs into our Bayesian networks.

From the trust management process in Chapter 6 we identified the entities to be trusted, risks to avoid, the required confidence levels and the associated beliefs into a trust policy. This trust policy will now be used in order to expand the *Provide Filtering* and *Allow Filtering* Bayesian networks to handle trust.

7.1 Provide Filtering

The Provide filtering scenario with the new information can be seen in Figure 7.1.

Comparing this to the Bayesian network in the second stage (see Figure 5.1) we now take the honesty and competence of the requester into consideration when receiving information from that entity, according to Section 2.4.1. This is done by specifying the probability of being Honest by stating a value of the honesty belief (ranging from *low* to *high*, with 5 states in total) given that he *is* honest (from the *Honest?* node), and the same in the case where he is dishonest. In the honesty node the associated conditional probability table shows the following:

$$P(Honesty|Yes) \text{ and}$$
$$P(Honesty|No)$$

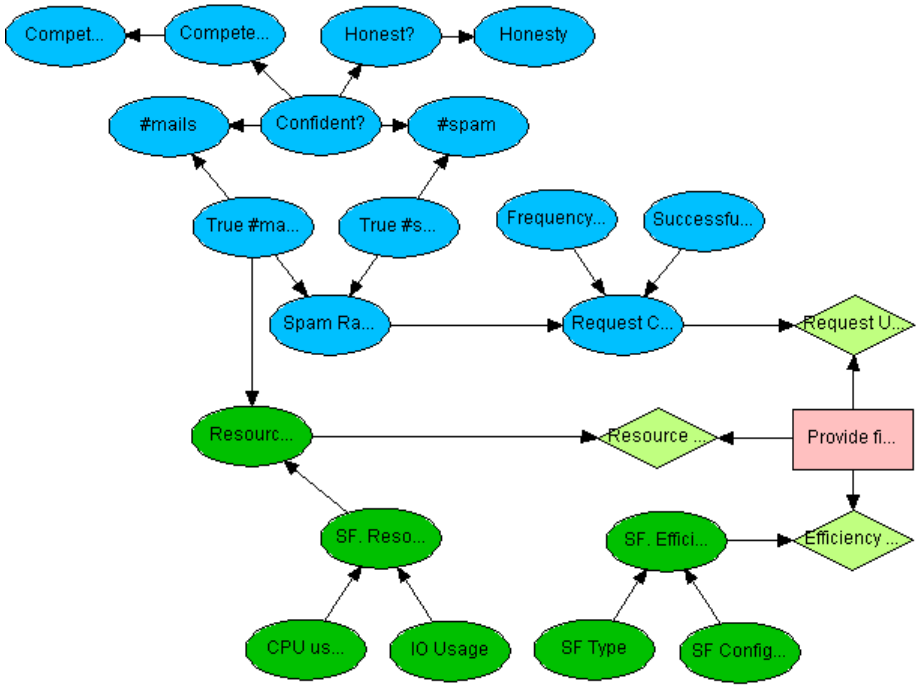

Figure 7.1: Bayesian network for *Provide Filtering* with trust. SF is short for Spam Filter.

This also applies for his competence. With this information it will now be possible to evaluate whether the risks (in this case *Inaccurate information* and *False requests*, according to Section 6.3.1) should be accepted. In the second stage we only looked at #mails. #spam is also considered in order to know what the spam rate is. The spam rate is, together with information regarding frequency of requests and successful requests, used in order to establish a request confidence level so it can be determined whether the request is worth the resources required to process it. This is modeled according to Section 2.4.1 where information/facts describing some entity should be modeled as shown in Figure 2.2. #spam and #mails are determined by the two beliefs: honesty and competence in the entity providing the information. The *Provide Filtering* Bayesian network has the following new nodes:

- **Confident?:** Information from other sources, in this case the requester, should be evaluated by considering the confidence in the source. The confidence is based on measurements on competence and honesty and is thus connected to the *Competent?* and *Honest?* nodes. The node has two states: *Yes* and *No*.
- **Competent?/Competence:** Competence is indicated by the node Compe-

tent? with the states *yes* and *no*. The Confident? node will receive information stating the probability for *yes* and *no* whether the source is competent. In the Competence node a competence-level is selected, ranging from *low* to *high* ranging over 5 states, each with a different distribution for *yes* and *no*. For instance the competence-level *low* could give the distribution: Yes: 10%, No: 90%, depending on the specific situation.

- **Honest?/Honesty:** The Honest? and Honesty nodes operate similar to the *Competent?/Competence* nodes just mentioned. In this case the nodes state how honest the source is.

- **Request Confidence Level:** When a request is received it should also be evaluated for whether it is bogus or a real request. In order to conclude this, information regarding spam rate, frequency of requests, and successful requests is needed, and is thus connected to the *Spam Rate*, *Frequency of requests*, and *Successful requests* nodes. The node has 5 states: Low, Low-Medium, Medium, Medium-High, and High.

- **Spam Rate:** The reason why this information is needed is in order to decide whether or not the requesting machine is entitled to request filtering. The spam rate is measured by number of e-mails and number of spams received and is thus connected to the *True #mails* and *True #spam* nodes. The node has 3 states: Low, Medium, and High.

- **#mails:** Even though this node was present in stage two it has changes radically, since it is now connected to the *Confident?* and *True #mails* nodes. This makes it possible to specify how the confidence influence the probability for #mails to be in a given state. The conditional probability table for the *#mails* node can be seen in Table 7.1.

True #mails	**Low**		**Medium**		**High**	
Confident?	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**
Low	1	1	0	1	0	1
Medium	0	1	1	1	0	1
High	0	1	0	1	1	1

Table 7.1: CPT for #mails

- **#spam:** Number of spam e-mails received by the requester. This value should also be evaluated according to the *Confident?* and *True #spam* nodes. The node has 3 states: Low, Medium, and High.

- **True #mails/#spam:** These nodes indicate the true value of the amount of e-mails and spam. When deciding the spam rate it is by the use of these nodes that the rate is determined. These nodes have 3 states: Low, Medium, and High.

- **Frequency of requests:** The frequency of requests is a measurement of how many requests the given machine has made within some time period. The node has 3 states: Low, Medium, and High. High frequency will have negative impact on the request confidence level.

- **Successful requests:** Some requests may be accepted after which the requester might decide to cancel the request thus wasting resources. This is not desirable and will therefore have a negative impact on the request confidence level. The node has 3 states: Low, Medium, and High.

- **Request Utility:** This utility node specifies the cost or gain based on the state of the *Request Confidence Level* node.

7.2 Allow Filtering

The Allow filtering scenario with trust management incorporated can be seen in Figure 7.2.

In order to avoid the threats of getting inaccurate information, private information revealed, and Inefficient spam filter, Honesty has been added to the Bayesian network in Figure 7.2. This honesty is used when measuring False Positives and False Negatives, and the correctness of the information regarding the spam filter efficiency. This has been added according to Section 2.4.1 by adding an honesty node affecting the nodes concerning information from other sources, in this case: False Negatives(FN), False Positives(FP), SF Type, and SF Configuration. False positives and False negatives are also affected by a test indicating the correctness of the information provided from the receiver. Another thing that has been added is the use of the ability to ask others, where a third party is asked for advice. Nodes involved in this part is colored orange. In this case another machine is asked for its opinion on the true values of False Positives and False Negatives. The information it delivers is trusted/believed according to the Honesty and Competence we have in it. The following explains the new nodes:

- **Confident? (other/orange):** Information from other sources, in this case the another machine, should be evaluated by considering the confidence in the source. The confidence is based on measurements on competence and

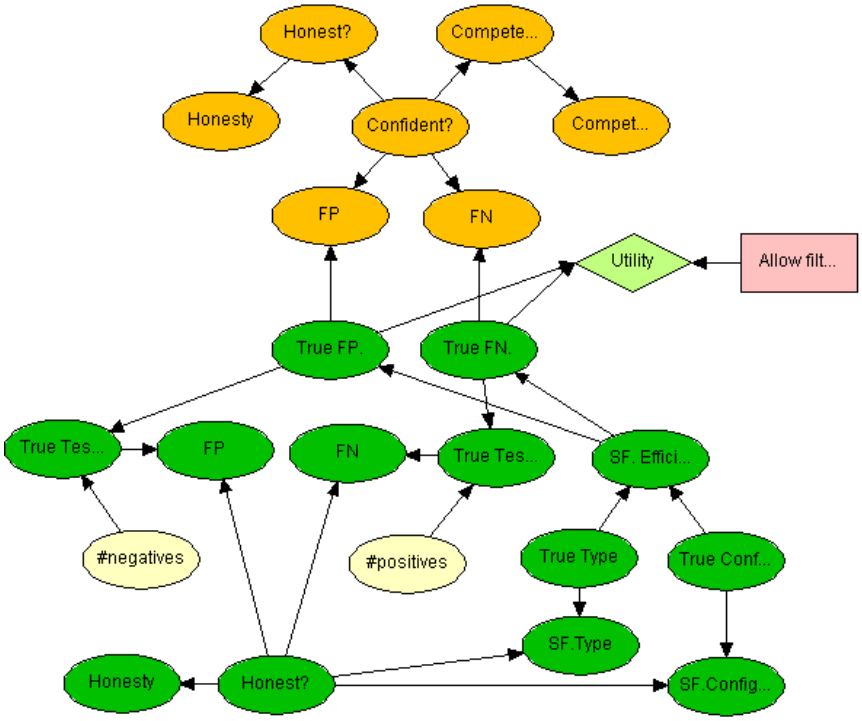

Figure 7.2: Bayesian network for *Allow Filtering* with trust. SF is short for Spam Filter, FP is short for False Positives, and FN is short for False Negatives.

honesty and is thus connected to the *Competent?* and *Honest?* nodes. The node has two states: *Yes* and *No*.

- **Competent?/Competence (other/orange):** Similar to the *Competent?/Competence* nodes in the *Provide Filtering* scenario. These nodes concern the other machine asked.
- **Honest?/Honesty (other/orange):** Similar to the Honest?/Honesty nodes in the *Provide Filtering* scenario.
- **FP/FN (other/orange):** These are the results delivered by the other machine asked. The nodes have four states: Many, Some, Few and Very Few. They are evaluated by considering both the confidence in the other machine, and the true values of the false positives and false negatives and are thus connected to those nodes.
- **FN/FP:** The measurement of false positives and false negatives is a result of

a test (see Section 5.1) where some sample e-mails (containing both spam and non-spam) are tested on the machine's spam filter. Although these nodes are present in the Bayesian network presented in the second stage (see Chapter 5) they are now evaluated by examining the honesty of the entity and their conditional probability table has changed accordingly. They are also affected by the true test results and are thus connected to the *True Test FP*, *True Test FN* and *Honesty?* nodes. The nodes have four states: Many, Some, Few and Very Few.

- **True Test FP/FN:** These nodes represent the true results of the spam filter test and are affected by the size of the e-mail collections on which the test was performed. Thus they are connected to the *#negatives* and *#positive* nodes. The nodes have four states: Many, Some, Few and Very Few.
- **Honest?/Honesty:** Indicates the honesty-level of the machine requested to do spam filtering (the receiver). Similar to the *Honest?/Honesty* nodes of the *Provide Filtering* scenario. This honesty measure is used when information from the receiver is used.
- **SF Type/SF Configuration:** These two nodes have also been evaluated according to the honesty and are thus connected to the *Honesty?* node. The resulting conditional probability table for SF Type is shown in Table 7.2.

True Type	**Type 1**		**Type 2**		**Type 3**		**Type 4**		**Type 5**	
Honest?	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**
Type 1	1	1	0	1	0	1	0	1	0	1
Type 2	0	1	1	1	0	1	0	1	0	1
Type 3	0	1	0	1	1	1	0	1	0	1
Type 4	0	1	0	1	0	1	1	1	0	1
Type 5	0	1	0	1	0	1	0	1	1	1

Table 7.2: CPT for SF Type.

- **True Type:** Even though the machine reports the type of spam filter it uses it might not be true. Therefore this node gives an approximation on the true value taking the measurement of the honesty of the receiver into consideration. The node has the same states as *SF Type*.
- **True Configuration:** As with spam filter type, the configuration specified might not be trustworthy information. Therefore a measurement of the true configuration is needed, in this case also determined by the honesty in the receiver. The node has the same states as *SF Configuration*.

As mentioned in this situation a third party is asked for advice. In this case, as shown in Figure 7.2, asking someone else is done without considering that it might have a cost. In the real world it might very well have some associated cost, maybe not money but at least time or doing some service in return. Handling this can be done by treating the case whether or not to ask another similar to a metric, using the structure mentioned in Section 2.4.1 for modeling metrics. The resulting Bayesian network can be seen in Figure 7.3.

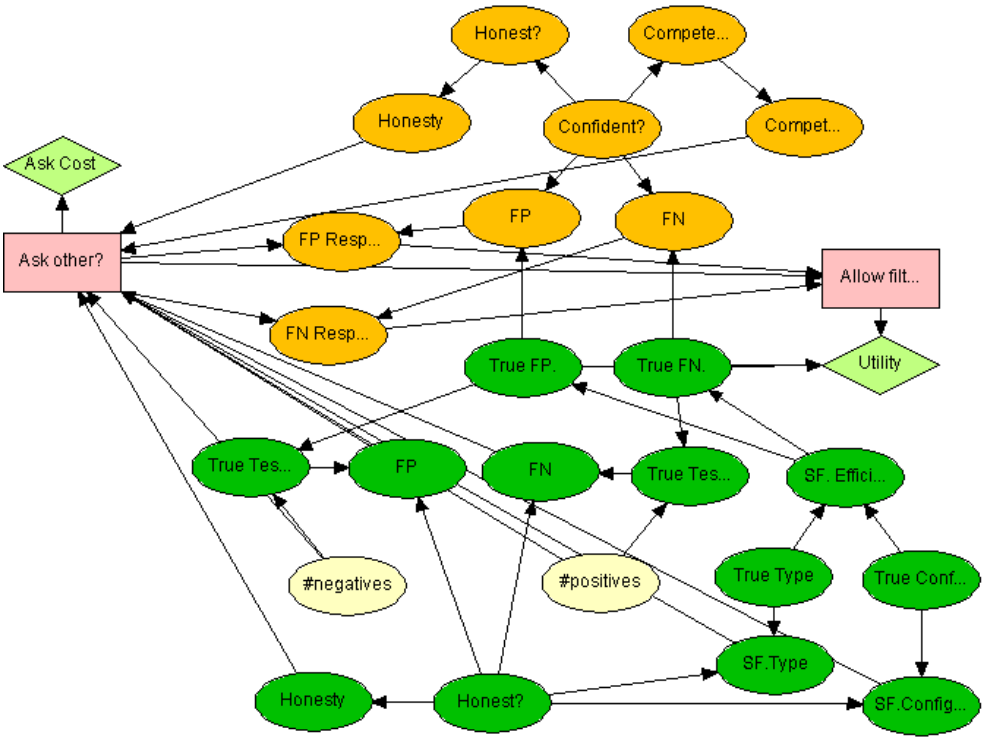

Figure 7.3: Bayesian network for *Allow Filtering* with trust. Evaluating if the third party should be asked for advice. SF is short for Spam Filter, FP is short for False Positives, and FN is short for False Negatives.

The many connections to the *Ask Other?* decision node makes the network very computational expensive. So expensive in fact, that it is not practically usable by TrustOne. The solution to this problem is described in Section 8.1.3.

In Chapter 8 the Bayesian networks and the concepts behind TrustOne will be tested and evaluated thoroughly.

Chapter 8

Test/Evaluation

To test the communication protocol and Bayesian networks of TrustOne, a simple simulation of the environment where TrustOne would operate has been created. The main focus of the simulation is to test whether TrustOne really has the desired properties described in the previous chapters. The simulation will be rather crude since its only purpose is to test the main features of the communication protocol and the structure of the Bayesian networks used in TrustOne.

8.1 Design

The two main elements of TrustOne that need to be simulated are the communication protocol and the two Bayesian networks. Thus we need to be able to simulate a network of machines, where the machines are able to communicate with each other.

Every machine in the network must be able to send, receive, and process e-mails. Also, when appropriate, each machine must be able to use the Bayesian networks to help making its decisions.

Each machine also needs to understand and obey the communication protocol described in Chapter 5.

8.1.1 The Network

The network chosen for the simulation can be seen in Figure 8.1.

This structure was chosen because it has many different network qualities. Some machines have many connections to others machines like machine 8 and 12, while others have only a few connections like machines 7 and 14. The network

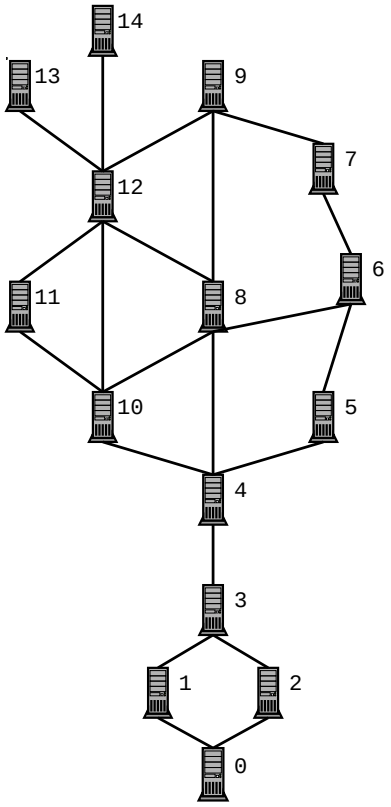

Figure 8.1: The structure of the network of machines used in the simulations.

also has a critical connection like the one between machine 3 and 4, where the traffic from a large part of the network is tunneled through a few machines.

With this network structure we should be able to test TrustOne on network structures similar to those used in the real world.

If a machine is to be able to send messages to other machines, it needs some information about its environment. It needs to know which other machines exist and which machines are its neighbors. Since a machine is not directly connected (a neighbor) to all the other machines in the network, it cannot send messages directly to all machines. Some messages must be sent through others. Thus a machine needs to know to which other machine it must send a message even if the machine does not have a direct connection to the destination of the message.

To this end a *route table* has been created (see Table 8.1).

By using this route table each machine is able to send a message to any other

machine/machine	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	0	1	2	2	1	2	1	2	1	2	1	2	1	2	1
1	0	1	0	3	3	3	3	3	3	3	3	3	3	3	3
2	0	0	2	3	3	3	3	3	3	3	3	3	3	3	3
3	1	1	2	3	4	4	4	4	4	4	4	4	4	4	4
4	3	3	3	3	4	5	8	5	8	8	10	10	10	8	8
5	4	4	4	4	4	5	6	6	4	6	4	4	6	4	6
6	5	8	5	8	5	5	6	7	8	7	8	8	8	8	8
7	6	9	6	9	6	6	6	7	9	9	6	9	9	9	9
8	4	4	4	4	4	4	6	6	8	9	10	10	12	12	12
9	8	8	8	8	8	7	8	7	8	9	12	12	12	12	12
10	4	4	4	4	4	4	8	8	8	12	10	11	12	12	12
11	10	10	10	10	10	10	10	12	10	12	10	11	12	12	12
12	8	10	8	10	8	8	8	9	8	9	10	11	12	13	14
13	12	12	12	12	12	12	12	12	12	12	12	12	12	13	12
14	12	12	12	12	12	12	12	12	12	12	12	12	12	12	14

Table 8.1: The route table for the simulated network of machines. Each row contains the information required by one machine. Thus if machine 10 must deliver a message to machine 2, it simply delivers the message to machine 4. Machine 4 will now deliver the message to machine 3 who in turn will deliver the message to machine 2.

machine without having to know the structure of the network. It only needs to know in which general direction to send a message and then let the other machines deliver the message the rest of the way to its destination.

In the simulation the machines do not run simultaneously as they would in the real world, but rather they run sequentially. One at a time in numerical order starting from machine 0. A *turn* is when all machines have run once. Thus after 100 turns each machine has run 100 times.

8.1.2 The Machines

As mentioned earlier, each individual machine must be able to handle all the messages sent by TrustOne. They must also be able to send and receive e-mails, and to run these e-mails through a spam filter. We here present a brief description of how a machine is simulated/implemented.

Each machine has 2 main 'loops', an outer and an inner loop. The inner loop controls what happens when an e-mail is received while the outer loop handles everything else.

We start with the inner loop. When an e-mail is received, one of the following things happen depending on the type of e-mail. Only the most important actions are described, several smaller ones have been omitted to keep the description simple.

The *stop* command indicates that the execution ends.

- *Normal e-mail:* A 'normal' e-mail is any e-mail that is not used by TrustOne. The normal e-mail might be spam.

```
If e-mail was delivered by a machine running spam filter for me
   If e-mail not for me
      Forward e-mail
   else
      Remove e-mail from network
   Stop
If this machine is running a spam filter
   If the e-mail is for me
      Run e-mail through spam filter
      Store result
      Remove e-mail
      Stop
   else
      If the e-mail is to be delivered to a machine
       for which I am running a spam filter
         Run e-mail through spam filter
         Store result
         If e-mail is spam
            Remove e-mail
            Stop
Forward the e-mail
```

- *Request e-mail:* The request e-mail is the e-mail sent when one machine requests another to run a spam filter on its behalf, that is, it is the receiver who must process this e-mail.

```
receiver:
   If this machine does not accept requests
      Stop
   If requester sends suggestion
      Run 'provide' BN using requesters suggestion
         If the BN suggests that the utility is negative
            Run 'provide' BN using own suggestion
               If the BN suggests that the utility is negative
                  Send a Reject request message to the requester
                  Stop
               else
                  Run the two e-mail collections supplied by the
                   requester through the spam filter
                  Send a request response message to the requester
                  Stop
         else
            Start running spam filter on requesters behalf
```

```
            Send accept request message
   else
      Run 'provide' BN using own suggestion
      If the BN suggests that the utility is negative
         Send a Reject request message to the requester
         Stop
      else
         Run the two e-mail collections supplied by the
          requester through the spam filter
         Send a request response message to the requester
         Stop
```

- *Request response e-mail:* The request response e-mail is sent as a reply to the request message, in the case where the request could not be accepted in its current form. The request response message contains a counter-suggestion from the receiver to the requester.

```
requester:
   Run 'allow' BN using the data from the response
      If the BN suggests that the utility is negative
         Send a reject suggestion message to the receiver
         stop
      else
         Send an accept suggestion message to the receiver
         Stop checking e-mails delivered by that machine
```

- *Accept request e-mail:* If the receiver finds the request acceptable, it responds by sending an accept request message to the requester.

```
requester:
   Stop checking e-mails delivered by the receiver for spam
```

- *Reject request e-mail:* If the receiver finds the request unacceptable and it does not have an acceptable counter-suggestion, then it responds to the request by sending a reject request message to the requester.

```
requester:
   Continue checking e-mails delivered by the receiver for spam
```

- *Accept suggestion e-mail:* If the requester accepts the counter-suggestion proposed by the receiver, it responds with an accept suggestion message.

```
receiver:
   Start filtering e-mails to the requester through a spam filter.
```

- *Reject suggestion e-mail:* If the requester rejects the counter-suggestion from the receiver, it responds with the reject suggestion message.

```
     receiver:
        Do nothing
```

The outer loop is a bit simpler than the inner loop. It is simply:

```
While the machine has unprocessed e-mails waiting
   Run the inner loop
Send (normal) e-mails if thus desired
If this machine is running a spam filter
   For each neighboring machine
      If (number of spam) / (total number of e-mails) is too high
         Send a request to that neighbor
```

As mentioned earlier, this pseudo-code does not contain all the actions taken by the receiver and the requester. For instance, both machines keep statistics about the other machine that will help them in their decisions.

The requester keeps track of:

- The number of e-mails and spam that are delivered by the receiver
- The requester's evaluation of the honesty and competence of the receiver

and the receiver keeps track of:

- How many requests it has received from the requester and how many of these requests were successful
- The receivers evaluation of the honesty of the requester

Both receiver and requester must of course keep track of

- Which machines are running spam filters on its behalf, and which machines it is providing spam filters for

This was a brief description of the implementation of the simulation of TrustOne. We will now take a look at the simulations themselves and the results.

8.1.3 The Bayesian Networks

The Bayesian networks have been implemented just as they are described in Chapter 7 with one exception. The *Allow Filtering* network shown in Figure 7.3 is not used in the shown form. The many connections to the *Ask Other?* decision node make the network very computational costly. Far too costly to be used in TrustOne. Thus changes to the network have to be made in order for the network to be useful.

In Figure 7.3 the *Ask Other?* decision node is used to decide whether there will be an increase in the expected utility of the *Allow Filtering* decision node, if another machine is asked for advise. This increase will also have to cover the cost of *Ask Other?*.

There is another way to gain the information of whether to ask other or not. We can use the *value of information* technique[6] to make a prediction whether the information we desire, is valuable enough to increase our expected gain when making the *Allow Filtering* decision, despite the cost of getting this information.

Whether we should ask other depends on what the answer is going to be. If the answer can affect the *Allow Filtering* decision, then the answer could be important. If on the other hand, the outcome of the answer will have no effect on the *Allow Filtering* decision, then there is no point in asking. What we need is to examine the possible answers we can receive if we decide to ask other and the expected utilities given those answers. To be more precise, what we need to know is if:

$$EU(ask) - EU(don't\ ask) > cost(asking)$$

where *EU(ask)* is the expected utility if we decide to ask other and *EU(dont ask)* is the expected utility if we don't ask other. *cost(asking)* is the cost we have to pay if we decide to ask other. Only when this expression is true is it desirable to ask other.

To calculate *EU(ask)* all the possible answers that we can receive from the *other* machine and how likely we are to receive these answers must be considered. If for instance the answer could be *yes* or *no*, the *EU(ask)* is calculated as:

$$EU(ask) = P(Yes)EU(Yes) + P(No)EU(No)$$

In the *Allow Filtering* network, two answers are received if it is decided to ask other. Evidence is received for both the *FP* and *FN* nodes of the other machine (the orange nodes in Figure 7.3) and each of these nodes has four states as described in Section 7.2. All combinations of the two answers must be considered. Thus we get:

$$EU(ask) = \sum_{0<i\leq 4} \sum_{0<j\leq 4} (P_{FP}(stage_i) \times P_{FN}(stage_j))EU(FP_{stage_i}, FN_{stage_j}) \tag{8.1}$$

The required values of Equation 8.1 is easily retrieved using the *Allow Filtering* network (see Figure 7.2) itself. The $P_{FP}(stage_i)$ and $P_{FN}(stage_i)$ is found by entering all the evidence in to the *Allow Filtering* network except into the *FP* and *FN* nodes for the other machine (the orange nodes). *EU(don't ask)* is now the value of the *Allow Filtering* utility node.

The $EU(FP_{stage_i}, FN_{stage_j})$ is now easily found by inserting evidence for each possible combination of the *FP* and *FN* nodes for the other machine, propagate, and read the *Allow Filtering* utility for each combination.

This value of information approach requires that the *Allow filtering* network shown in Figure 7.2 be run once for each combination of the states of the *FP* and *FN* nodes for the other machine[1], but it is still a lot more computational efficient than running the *Allow Filtering* network shown in Figure 7.3 just once. Also the value of information calculation is only performed when a machine has received a request response message, which keeps the required computational cost very low.

8.2 Simulation Results

To test TrustOne the simulations have been split into 3 separate tests. The first tests concentrate on the communication protocol used by TrustOne. The second set of tests concentrate on the *Provide Filtering* and *Allow Filtering* Bayesian networks. Finally the last tests will examine the behavior of TrustOne as a whole and compare the results with ordinary client-side filtering.

8.2.1 The Communication Protocol Tests

We will start by testing the main properties of the TrustOne communication protocol. This is easily done by giving all the 15 machines in the simulation the same very efficient spam filter. The good spam filters make it easy for each machine to identify spam and the source of the spam. Also, all the machines will accept any requests to run a spam filter on another machines behalf. Thus we should expect that the machines who sends the spam will be encircled by machines who run spam filters on all e-mails coming from the spammer. The machines who are not directly connected to the source of the spam should not be running any spam filters.

We simulate three differing situations, starting with a single spammer and a single machine running a spam filter. Machine 0 is the designated spammer and will send spam e-mails to machine 14 which in this case is the only end user[2]. Only machine 14 will be running a spam filter. The rest of the machines can be considered as routers in the network, see Figure 8.1.

A trace of the simulation is as follows[3]:

```
Successful initialization of machines
```

[1] 16 times in our case

[2] An end user is a real person and not some automated software. As mentioned in Section 5.1, only the receiver of an e-mail can classify it as spam or not spam with 100% accuracy.

[3] The trace can also be seen in Appendix A.

```
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14's behalf
Machine: 14, does not reviece a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 12
Machine: 8 is now running a spam filter on machine: 12's behalf
Machine: 4 has received a request from machine: 8
Machine: 4 is now running a spam filter on machine: 8's behalf
Machine: 3 has received a request from machine: 4
Machine: 3 is now running a spam filter on machine: 4's behalf
Machine: 1 has received a request from machine: 3
Machine: 1 is now running a spam filter on machine: 3's behalf
Machine: 0 has received a request from machine: 1
Machine: 12, does not receive a lot of spam. Shutting down spam filter
Machine: 8, does not receive a lot of spam. Shutting down spam filter
Machine: 4, does not receive a lot of spam. Shutting down spam filter
Machine: 3, does not receive a lot of spam. Shutting down spam filter
End of simulation...
Machines currently running spam filters: 1,
```

As it can be seen from the route table (see Table 8.1), the trace follows the exact path of the spam e-mails from machine 0 to machine 14. As a result the spammer, machine 0, has been isolated from the rest of machines in the network by machine 1. Just as intended.

In this first test we only had one end user. This is not very realistic thus we run the test again with machines: 3, 5, 6, 7, 9, 11, 13 , and 14 as end users. The resulting simulation trace can be seen in Appendix B. This time machines 1 and 2 isolate the spammer, machine 0.

Machine 0 is rather easily isolated because of its placement in the simulated network. As a final test of the TrustOne protocol we thus extend the previous tests by letting a machine with many connections to other machines send spam. Both machine 0 and 8 will now send spam to the other end users. The full trace can be seen on Appendix C. After the simulation the spamming machines have been isolated by the machines 1, 2, 4, 6, 9, 10, 12. Just as expected.

8.2.2 The Bayesian Network Tests

To test the *Provide Filtering* and *Allow Filtering* networks used in TrustOne the networks were loaded into the program *Hugin Researcher*[12]. Evidence was then inserted and removed from each single node and the effect observed. Both the networks behaved as desired.

8.2.3 The TrustOne System Tests

At this the final stage of testing, both the communication protocol and the Bayesian networks are tested as a whole. We will reuse the setup of machines used on the testing of the communication protocol, and only change the input to the Bayesian networks. We should then be able to see any changes in the result caused by the input changes.

We start by having machine 0 as the spammer and machine 14 as the end user, just as in the first test of the communication protocol. This time however we let machine 8 use a spam filter that is less efficient than the filter used by the other machines. All machines will classify e-mails with a 99% accuracy except machine 8 who only has 80% accuracy.

A trace of the simulation is as follows[4]:

```
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14's behalf
Machine: 14, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 12
Machine: 12 has received a request response from machine: 8
Machine: 12 is asking machine: 13
Machine: 8 has received reject suggestion from machine: 12
End of simulation...
Machines currently running spam filters: 12
```

As it can be seen, machine 12 request machine 8 to run a spam filter on its behalf, but machine 8 cannot run the very efficient spam filter that machine 12 suggests in the request. By using the *Provide filtering* Bayesian network Machine 8 however believes that its lesser efficient spam filter will still be beneficial and thus sends a counter-suggestion back to machine 12. Machine 12 inputs the counter-suggestion into its *Allow Filtering* Bayesian network, which finds that it is worth the effort to ask the advice of another machine. In this case machine 13. Based on its own evaluation and machine 13's advice, it concludes that Machine 8's offer is not good enough. Machine 12 therefore decides to send a reject suggestion message to machine 8 and keep running its own spam filter.

As this simulation shows, the 'chain' of requests will stop if the receiver of the request cannot fulfill the job of classifying e-mails to the requesters satisfaction.

If we rerun the simulation but with machines: 5, 6, 7, 9, 11, 13, and 14 as end users and still having machine 8 running the lesser efficient spam filter we get the trace shown in Appendix E. As the trace shows, machine 0 is isolated by machines 1 and 2. This time the isolation worked better than in the last simulation. Even though machine 12 is in the same situation where it cannot trust machine 8 to filter

[4]The trace can also be seen in Appendix D.

spam on its behalf, other machines who are not dependent on machine 8 will get the requests to machine 1 and 2. As soon as machine 0 is isolated, machine 12 will no longer receive spam and can therefore shut down its own spam filter. As this simulation shows, even if some machines are not able to perform the task of spam filtering, TrustOne can still perform well.

If instead of reducing the efficiency of machine 8's spam filter, we reduced its competence and honesty in the eyes of the other machines, almost exactly the same scenario would occur as in the previous simulation. The communication protocol only cares whether one machine accepts another machines request or not, not the reason for the rejection or acceptance. Thus we will not make further such simulations.

8.2.4 Comparison

The main purpose of TrustOne is to save some of the resources currently wasted on spam. Now that we have tested that TrustOne behaves as predicted, we also need to examine whether it actually saves any resources.

To examine resources saved by TrustOne, two simulations are run. One where the machines make use of TrustOne and another that does not. All the machines in the simulated network (see Figure 8.1) will be end users who run spam filters except machines 0 and 9 who are the designated spammers. All the machines (except the spammers of course) will be using very efficient spam filters to keep the complexity of the simulation results low.

The simulation software measures the following information:

- *Nr. of 'turns':* Shows the number of sequential updates[5] of machines in the current simulation.
- *Total nr. of e-mails:* Gives the number of e-mails that were created and sent during the simulation.
- *Total nr. of spam:* Gives the number of spam that were created and sent during the simulation.
- *Total nr. of sends:* Shows the total number of e-mails, spam and TrustOne messages sent.
- *Total nr. of forwards:* Shows the number of times a message has been forwarded from one machine to another.

[5] Described in Section 8.1.1.

- *Total nr. of spam filter checks:* Gives the total number of spam filter checks during the simulation.
- *Total nr. of spam filter checks where the e-mail was in fact spam:* Shows the number of times an e-mail was checked for spam and the e-mail was indeed spam. This number might be smaller than the number of sent spam since the spammers also send spam to each other and the spammers do not run spam filters.
- *Total nr. of successful spam filter checks:* Gives the number of times the spam filters made a correct classification.
- *Total nr. of requests/accepts/...:* These last five values show how many messages were sent of each of the message types[6] used by TrustOne.

We first run the simulation where the machines do *not* make use of TrustOne. The result of the simulation is:

```
Machines running spam filters at end of simulation:
    1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14,

Nr. of 'turns': 100000
Total nr. of e-mails: 130124
Total nr. of spam: 200002
Total nr. of sends: 330126
Total nr. of forwards: 664156
Total nr. of spam filter checks: 297499
Total nr. of spam filter checks where the e-mail was in fact spam: 185830
Total nr. of successful spam filter checks: 294525
Total nr. of requests: 0
Total nr. of accepts: 0
Total nr. of accept suggestions: 0
Total nr. of rejects: 0
Total nr. of reject suggestions: 0
```

Then the second simulation that is exactly like the first except the machines now make use of the TrustOne system. The result is:

```
Machines running spam filters at end of simulation:
    1, 2, 7, 8, 12,

Nr. of 'turns': 100000
Total nr. of e-mails: 130158
Total nr. of spam: 199999
Total nr. of sends: 330194
```

[6] As described in Section 5.1 and Section 8.1.2.

```
Total nr. of forwards: 234350
Total nr. of spam filter checks: 277500
Total nr. of spam filter checks where the e-mail was in fact spam: 193424
Total nr. of successful spam filter checks: 274648
Total nr. of requests: 21
Total nr. of accepts: 16
Total nr. of accept suggestions: 0
Total nr. of rejects: 0
Total nr. of reject suggestions: 0
```

Even though the exact number of e-mails and spam varies a little in the two simulations, the results are clear.

- *Total nr. of forwards:* The number of times an e-mail is forwarded is severely reduced. This is a direct consequence of the spam being removed by the ring of machines surrounding the spammers. The spam will no longer have to be delivered all the way to its destination.

- *Total nr. of spam filter checks:* The number of spam filter checks is also reduced. Machines no longer need to check e-mails for spam if the e-mails are delivered by a machine running a spam filter on their behalf.

- *Total nr. of spam filter checks where the e-mail was in fact spam:* Even though the number of spam filter checks has decreased the number of checks where the e-mail was in fact spam has increased. Thus every time a spam filter checks an e-mail there is greater chance that the e-mail is in fact spam, if the machines uses TrustOne. Thus the resources used by spam filters are used more efficiently. This result is not surprising since only the machines who have direct contact with the spammers will be running spam filters.

- *Total nr. of successful spam filter checks:* The number of successful spam filter checks has also dropped, but this is a direct result of the drop in overall spam filter checks.

- *Total nr. of requests/accepts/...:* The last five values show that 21 requests were sent and 16 of them were accepted. The 'missing' 5 requests were sent to the two spammers who do not respond to requests.

The structure of the simulated network and the placement of the end users and the spammers affect how many resources TrustOne will be able to save, but in every simulation TrustOne saves resources. Thus it can be concluded that TrustOne has the desired properties.

One weakness in TrustOne is that even though the total amount of resources required is reduced, the resources that *are* used must be spent by fewer machines.

Thus reducing the workload on most machines while at the same time increasing the workload considerably on the few machines close to the spammers.

Chapter 9

Future Work

In this report we have described a system making outsourcing of spam filtering on multiple machines possible thus saving resources. Several things could improve it and make it easier to use. This chapter will describe improvements and future work that could improve TrustOne.

9.1 Tool for Easing Bayesian Network Creation

Developing a Bayesian network based on information from a trust policy might not always be easy. In our case though, it is the process of adjusting the conditional probability tables according to the specific situation that will keep many people from using TrustOne. If this process could be automated it would make many people look at the system with less fear. Automating this process is actually not impossible since every detail regarding the given decision is identified in the trust policy. As the required confidences in the different entities and the likelihood and consequences of the different risks are also identified it should be possible to create the conditional probability tables.

What is needed is therefore a tool in which details from the trust policy can be specified. With this knowledge it will be possible to create the conditional probability tables for nodes by specifying the required confidence. For example as the required confidence in SF Type in the *Allow Filtering* network is decided, and that the belief: *Honesty* is used to describe this confidence in the entity: *Own Spam filter*, the conditional probability table can be created easily. If the required confidence is for instance High, then in the conditional probability table only when the belief is High no ambiguity should be present. If the belief is measured to be in state Low, the confidence of the information being described should be accordingly low, which therefore should be illustrated in ambiguous results in the conditional

probability table. When all conditional probability tables are defined the only thing left for the user is to adjust the utility nodes according to the desired outcome in the given situation.

9.2 Merging of Requests

When a machine runs spam filters on behalf of other machines, problems can arise when this machine wants to send a request. As an example see Figure 9.1. Assume that machine 4 is running spam filters for machine 1 and 3. Machine 4 has agreed to run a very strict spam filter on the behalf of machine 1 while it only runs a weak spam filter on machine 3's behalf. It has still not received any requests from machine 2.

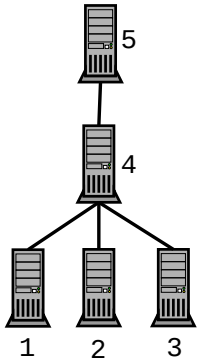

Figure 9.1: Sample network, showing potential merging of request problem.

Machine 4 now discovers that it receives a lot of spam from machine 5, and it would like to have machine 5 run a spam filter on its behalf. All e-mails to machines 1, 2, and 3 passes through machine 5 and are thus affected by what spam filter machine 5 uses. Lets assume that machine 5 accepts the request and thus allows machine 4 to shut down its own spam filters. What type and configuration of spam filter should machine 4 suggest to machine 5?

If machine 5 runs a very strict spam filter, then e-mails to machine 3 will be filtered using a much more strict spam filter than machine 3 initially agreed upon. Thus some e-mails that are not considered spam by machine 3, might be removed by the strict spam filter of machine 5.

If machine 5 runs a weak spam filter, then e-mails to machine 1 will no longer be filtered as strictly as initially agreed upon, thus machine 1 will most likely receive more spam.

Both solutions have undesirable results. A more thorough research into this problem could help optimize the results of TrustOne.

9.3 Billing System

Currently TrustOne relies on the willingness of machines to work for the good of others. The Machines that run spam filters on the behalf of others use their resources to make it possible for others to save their resources.

A better incentive will most likely be required. Something that could compensate for the resources used by the machines running spam filters. Some sort of billing system might be devised that allow the machines running spam filters to bill the machines that is now protected from spam for the service provided.

Other types of incentives could be researched and developed.

9.4 Tagging Spam with 'ADV'

Recently a law-proposal has suggested spam e-mails to be marked with an ADV in the subject field of the e-mail, thus spam as we know it today will be considered as a violation of the law[11]. Therefore theoretically the problem of classifying mails as spam or not spam would not be an issue any more as spam filters will be able to just remove e-mails with the ADV-tag. Realistically spam e-mails not marked with the ADV-tag will still be received, but the law would greatly reduce the amount of spam.

The question will be whether or not TrustOne still would be advantageous and should be determined by analyzing if resource waste is still a serious problem. E-mails marked with the ADV-tag do not require any special spam filter and would not require the whole trust-process in evaluating whether someone should be trusted to do the filtering since everyone would be able to do it perfect. The arguments for using TrustOne at first will not be as obvious any more. But as mentioned spam e-mails without this tag *will* be present and important to catch, just in a smaller amount, in which case TrustOne would be powerful in tracing the spammers sending illegal spam.

Chapter 10

Conclusion

In this report a spam filtering system called TrustOne has been developed. TrustOne consists of a communication protocol designed to let machines communicate information regarding spam filters. The focus has been on how to outsource who should be running spam filter and not on how the filtering process itself is being handled.

TrustOne consists of negotiations between machines in order to agree on whether spam filtering should be provided by the machine receiving the request. If this machine accepts to provide the service, a new decision should be made, namely whether or not this machine should be allowed to do the filtering. These two decisions have been modeled in two Bayesian networks; Provide filtering and Allow filtering.

By making it possible to outsource who should run spam filters, a problem of being able to trust other entities was identified. Letting others perform the spam filtering process on your behalf requires a great deal of confidence in the entities being trustworthy and capable of doing the task.

Developing the Bayesian networks was done in stages. The first editions of these did not handle the aspects of someone not being trustworthy, hence an analysis of trust management was done. This resulted in new Bayesian networks incorporating trusting beliefs such as honesty and competence in order to take the required trust aspect into account.

To verify that the communication protocol in TrustOne and the Bayesian networks behave as desired a simulation has been created. The simulation was a model of 15 machines interconnected using the TrustOne system. Some machines were end users and some were spammers. This simulation confirmed that both communication protocol and Bayesian networks were operating properly, resulting in isolating the source of spam thus saving resources, just as intended according to Section 1.1. The computational cost of using TrustOne showed to be of very little

concern, since the Bayesian networks are only used when processing requests.

Another requirement stated in Section 1.1 was that the spam filters should be able to filter spam efficiently and only filter what we consider spam. This has been achieved by incorporating trust management into the Bayesian networks making the decisions, thus only machines that perform within the acceptable limits are allowed to do filtering on our behalf.

Some problems are still not solved though, for instance when agreeing on type and configuration of a spam filter it can result in someone getting their e-mails filtered with a configuration less efficient than originally desired, which is a problem which needs further research.

We are aware that using TrustOne will not be seen as being very beneficial for some machines. It clearly shows that some machines will have to do more work, while others will have to do practically nothing, whereas today the workload is split pretty even among them when using ordinary client-side spam filters. If TrustOne is to function properly, it is important that every machine is willing to do a sacrifice for the common goal. At first this does obviously not sound appealing, but if accomplished everyone would eventually benefit from it.

Appendix A

Simulation Trace 1

Machine 0 is the designated spammer and will send spam to machine 14 which in this case is the only end user. Only machine 14 will be running a spam filter. The rest of the machines can be considered as routers in the network, see Figure 8.1. All machines have spam filters which are 99% efficient in classifying spam

```
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14’s behalf
Machine: 14, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 12
Machine: 8 is now running a spam filter on machine: 12’s behalf
Machine: 4 has received a request from machine: 8
Machine: 4 is now running a spam filter on machine: 8’s behalf
Machine: 3 has received a request from machine: 4
Machine: 3 is now running a spam filter on machine: 4’s behalf
Machine: 1 has received a request from machine: 3
Machine: 1 is now running a spam filter on machine: 3’s behalf
Machine: 0 has received a request from machine: 1
Machine: 12, does not receive a lot of spam. Shutting down spam filter
Machine: 8, does not receive a lot of spam. Shutting down spam filter
Machine: 4, does not receive a lot of spam. Shutting down spam filter
Machine: 3, does not receive a lot of spam. Shutting down spam filter
End of simulation...
Machines currently running spam filters: 1,
```

Appendix B

Simulation Trace 2

Machine 0 is the designated spammer and will send spam to machines: 3, 5, 6, 7, 9, 11, 13 , and 14 who are the end users. The rest of the machines can be considered as routers in the network, see Figure 8.1. All machines have spam filters which are 99% efficient in classifying spam

```
Machine: 12 has received a request from machine: 13
Machine: 12 is now running a spam filter on machine: 13's behalf
Machine: 13, does not receive a lot of spam. Shutting down spam filter
Machine: 4 has received a request from machine: 8
Machine: 4 is now running a spam filter on machine: 8's behalf
Machine: 4 has received a request from machine: 5
Machine: 4 is now running a spam filter on machine: 5's behalf
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14's behalf
Machine: 14, does not receive a lot of spam. Shutting down spam filter
Machine: 3 has received a request from machine: 4
Machine: 3 is now running a spam filter on machine: 4's behalf
Machine: 2 has received a request from machine: 3
Machine: 2 is now running a spam filter on machine: 3's behalf
Machine: 5, does not receive a lot of spam. Shutting down spam filter
Machine: 6, does not receive a lot of spam. Shutting down spam filter
Machine: 7, does not receive a lot of spam. Shutting down spam filter
Machine: 8, does not receive a lot of spam. Shutting down spam filter
Machine: 9, does not receive a lot of spam. Shutting down spam filter
Machine: 11, does not receive a lot of spam. Shutting down spam filter
Machine: 0 has received a request from machine: 2
Machine: 1 has received a request from machine: 3
Machine: 1 is now running a spam filter on machine: 3's behalf
Machine: 0 has received a request from machine: 1
Machine: 4, does not receive a lot of spam. Shutting down spam filter
Machine: 12, does not receive a lot of spam. Shutting down spam filter
Machine: 3, does not receive a lot of spam. Shutting down spam filter
```

```
End of simulation...
Machines currently running spam filters: 1, 2,
```

Appendix C

Simulation Trace 3

Both machine 0 and 8 will now send spam to the other end users, in this case machines: 3, 5, 6, 7, 9, 11, 13 , and 14. The rest of the machines can be considered as routers in the network, see Figure 8.1. All machines have spam filters which are 99% efficient in classifying spam

```
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14’s behalf
Machine: 14, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 6
Machine: 10 has received a request from machine: 11
Machine: 10 is now running a spam filter on machine: 11’s behalf
Machine: 6 has received a request from machine: 7
Machine: 6 is now running a spam filter on machine: 7’s behalf
Machine: 4 has received a request from machine: 5
Machine: 4 is now running a spam filter on machine: 5’s behalf
Machine: 12 has received a request from machine: 13
Machine: 12 is now running a spam filter on machine: 13’s behalf
Machine: 13, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 12
Machine: 8 has received a request from machine: 9
Machine: 4 has received a request from machine: 3
Machine: 4 is now running a spam filter on machine: 3’s behalf
Machine: 3 has received a request from machine: 4
Machine: 3 is now running a spam filter on machine: 4’s behalf
Machine: 8 has received a request from machine: 4
Machine: 2 has received a request from machine: 3
Machine: 2 is now running a spam filter on machine: 3’s behalf
Machine: 5, does not receive a lot of spam. Shutting down spam filter
Machine: 7, does not receive a lot of spam. Shutting down spam filter
Machine: 11, does not receive a lot of spam. Shutting down spam filter
Machine: 0 has received a request from machine: 2
Machine: 1 has received a request from machine: 3
```

```
Machine: 1 is now running a spam filter on machine: 3’s behalf
Machine: 3, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 10
Machine: 0 has received a request from machine: 1
End of simulation...
Machines currently running spam filters: 1, 2, 4, 6, 9, 10, 12,
```

Appendix D

Simulation Trace 4

Machine 0 is the designated spammer and will send spam to machine 14 which in this case is the only end user. Only machine 14 will be running a spam filter. The rest of the machines can be considered as routers in the network, see Figure 8.1. All machines have spam filters which are 99% efficient in classifying spam, except machine 8 who is only 80% efficient.

```
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14’s behalf
Machine: 14, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 12
Machine: 12 has received a request response from machine: 8
Machine: 12 is asking machine: 13
Machine: 8 has received reject suggestion from machine: 12
End of simulation...
Machines currently running spam filters: 12,
```

Appendix E

Simulation Trace 5

Machine 0 is the designated spammer. Machines: 5, 6, 7, 9, 11, 13, and 14 are end users. The rest of the machines can be considered as routers in the network, see Figure 8.1. All machines have spam filters which are 99% efficient in classifying spam, except machine 8 who is only 80% efficient.

```
Machine: 12 has received a request from machine: 14
Machine: 12 is now running a spam filter on machine: 14's behalf
Machine: 14, does not receive a lot of spam. Shutting down spam filter
Machine: 6 has received a request from machine: 7
Machine: 6 is now running a spam filter on machine: 7's behalf
Machine: 10 has received a request from machine: 11
Machine: 10 is now running a spam filter on machine: 11's behalf
Machine: 4 has received a request from machine: 5
Machine: 4 is now running a spam filter on machine: 5's behalf
Machine: 8 has received a request from machine: 6
Machine: 6 has received a request response from machine: 8
Machine: 6 is not asking another machine.
Machine: 8 has received accept suggestion from machine: 6
Machine: 8 is now running a spam filter on Machine: 6 behalf
Machine: 12 has received a request from machine: 13
Machine: 12 is now running a spam filter on machine: 13's behalf
Machine: 13, does not receive a lot of spam. Shutting down spam filter
Machine: 8 has received a request from machine: 9
Machine: 9 has received a request response from machine: 8
Machine: 9 is not asking another machine.
Machine: 8 has received accept suggestion from machine: 9
Machine: 8 is now running a spam filter on Machine: 9 behalf
Machine: 2 has received a request from machine: 3
Machine: 2 is now running a spam filter on machine: 3's behalf
Machine: 3, does not receive a lot of spam. Shutting down spam filter
Machine: 5, does not receive a lot of spam. Shutting down spam filter
Machine: 7, does not receive a lot of spam. Shutting down spam filter
```

```
Machine: 9, does not receive a lot of spam. Shutting down spam filter
Machine: 11, does not receive a lot of spam. Shutting down spam filter
Machine: 0 has received a request from machine: 2
Machine: 4 has received a request from machine: 8
Machine: 8 has received a request response from machine: 4
Machine: 8 is asking machine: 4
Machine: 4 has received accept suggestion from machine: 8
Machine: 4 is now running a spam filter on Machine: 8 behalf
Machine: 3 has received a request from machine: 4
Machine: 3 is now running a spam filter on machine: 4’s behalf
Machine: 8 has received a request from machine: 12
Machine: 12 has received a request response from machine: 8
Machine: 12 is not asking another machine.
Machine: 8 has received reject suggestion from machine: 12
Machine: 6, does not receive a lot of spam. Shutting down spam filter
Machine: 10, does not receive a lot of spam. Shutting down spam filter
Machine: 8, does not receive a lot of spam. Shutting down spam filter
Machine: 1 has received a request from machine: 3
Machine: 1 is now running a spam filter on machine: 3’s behalf
Machine: 0 has received a request from machine: 1
Machine: 4, does not receive a lot of spam. Shutting down spam filter
Machine: 3, does not receive a lot of spam. Shutting down spam filter
Machine: 12, does not receive a lot of spam. Shutting down spam filter
End of simulation...
Machines currently running spam filters: 1, 2,
```

Bibliography

[1] Vicomsoft, *White Papers - Spam Tutorial*, http://www.vicomsoft.com/knowledge/reference/spam_tutorial/email_spam_filter.html, 2003

[2] SpamAssasin, http://spamassassin.org

[3] Paul Graham, *A Plan for Spam*, http://www.paulgraham.com/spam.html, 2002

[4] Dean Povey, *Developing Electronic Trust Policies Using a Risk Management Model*, In Proceedings of the 1999 CQRE [Secure] congress, 1999

[5] Andrew S. Tanenbaum, *Computer Networks, Third edition*, Prentice Hall International, 1996

[6] Finn V. Jensen, *Bayesian Networks and Decision Graphs*, Springer-Verlag New York, 2001

[7] D. Harrison McKnight, Norman L. Chervany, and Larry L. Cummings, *Trust Formation In New Organization Relationships*, MISRC Working Paper, http://www.misrc.umn.edu/wpaper/WorkingPapers/9601.pdf, 1996

[8] D. Harrison McKnight and Norman L. Chervany, *The Meaning of Trust*, MISRC Working Paper, http://www.misrc.umn.edu/wpaper/WorkingPapers/9604.pdf, 1996

[9] An Atlas of Cyberspaces, http://www.cybergeography.org/atlas/geographic.html, 2003

[10] Mozilla.org, http://www.mozilla.org, 2003

[11] Spam laws, http://www.spamlaws.com, 2003

[12] Hugin Researcher, http://www.hugin.com, 2003

[13] ePrivacy Group, *Spam By the Numbers*,
http://www.eprivacygroup.com/pdfs/SpamByTheNumbers.pdf, 2003