

# **Logical Fibering Based Web Access Management**

zur Erlangung des akademischen Grades eines  
Doktors der Ingenieurwissenschaften

der Fakultät für Informatik  
der Universität Fridericiana zu Karlsruhe (TH)

**genehmigte**

**Dissertation**

von

**Marvin Oliver Schneider**

aus Münster/Westf.

Tag der mündlichen Prüfung: 16.07.2007

Erster Gutachter: Prof. Dr. Jacques Calmet

Zweiter Gutachter: Prof. Dr. Jochen Pfalzgraf

# Zusammenfassung

Diese Arbeit präsentiert einen auf Logischen Faserungen beruhenden Webzugangsmanagement-Ansatz zur Verhinderung von Denial of Service (DoS) und Distributed Denial of Service (DDoS)-Angriffen.

Sowohl Denial of Service als auch Distributed Denial of Service-Angriffe zielen auf das Aufbrauchen knapper Systemressourcen beim Opfer mit dem Ergebnis, dass berechtigten Benutzern ein Netzwerkdienst verweigert wird, wobei Distributed Denial of Service-Angriffe mehrere Netzwerkknoten mit geringer Sicherheit als Handler/Agenten ausnutzen, um die Angriffssituation so noch um ein Vielfaches zu verstärken.

Verschiedene aktuelle Lösungsansätze bieten nur sehr rudimentäre Möglichkeiten der Verteidigung: Einige behandeln auf eine übervereinfachte Art und Weise nur Teile der Symptome, andere konzentrieren sich auf den Zustand des Knotens, auf dem sie sich befinden, und lassen die Gefahr der eventuellen Blockierung legitimer Verbindungen ausser Acht. Weitere haben grosse Schwierigkeiten, Schlussfolgerungen aus der Angriffssituation zu ziehen.

Hauptursachen für alle diese Probleme sind – neben anderen – die weitverbreitete Verschleierung von Informationen durch Angreifer sowie das Bewirken von sehr starken negativen Auswirkungen über einen kurzen Zeitraum hinweg.

Der vorliegende Ansatz bietet eine Alternative durch die Implementierung des Systems „Fibered Guard“, einer intelligenten Personal Firewall, die erfolgreich Angriffe von legitimen Verbindungen unterscheiden kann. Dies wird auf empirische Art und Weise erreicht, indem der Systemstatus überwacht wird, sowie basierend auf den Vorzügen der logischen Faserungen in Verbindung mit einem Data Mining Algorithmus zur Erstellung von Schlussfolgerungen.

Eine logische Faserung ist ein abstrakter Faserraum  $\xi = (E, \pi, B)$ , in dem die Faser  $F$  über jeden Punkt  $b$  im Basisraum  $B$  eine logische Struktur ist. Das System „Fibered Guard“ benutzt gefaserte Faserungen für die zutreffende Darstellung der Daten einschliesslich der Reduzierung der Verbindungsdaten auf ihre Grundbausteine und der Möglichkeit, logische Strukturen direkt auf den Daten abzubilden. Die Verbindungsdaten werden in der primären Datendarstellung, die unverarbeitete Informationen eingehender Verbindungen enthält, sowie in einer sekundären Datendarstellung, die Regelsätze für die Blockierung von Angriffen zum Inhalt hat, gespeichert.

Der Data Mining Algorithmus wird zur Unterstützung der Erkennung von Zusammenhängen benutzt, ein Prozess, der an Mustererkennungsverfahren erinnert. Durch einen Klassifizierungsregel-Algorithmus werden Regeln für den normalen Operationsmodus sowie für den Angriffsmodus gefunden und auf der sekundären Datendarstellung abgebildet, die später für die intelligente Identifizierung von Angriffen über ein Vergleichsmodul genutzt wird.

Der Data Mining Algorithmus ist mit einem Künstlichen Neuronalen Netzwerk (Multilayer Perceptron) im Rahmen seiner Vorauswertung verglichen worden.

In realistischen Simulationen hat sich Fibered Guard als sehr erfolgreich gegen Denial of Service-Angriffe erwiesen, wobei drohende Angriffsverbindungen blockiert und legitime Verbindungen gehalten wurden. Sogar im Falle der viel komplexeren Distributed Denial of Service-Angriffe hat Fibered Guard eine sehr effiziente Verteidigung sowie eine zufriedenstellende Behandlung von legitimen Verbindungen gezeigt.

Eine weitere natürliche Entwicklung dieses Ansatzes wäre ein intelligentes System für die generelle Verteidigung gegen Denial of Service und Distributed Denial of Service-Angriffe in privaten, kommerziellen oder industriellen Anwendungen.

# Abstract

This work presents a Logical Fibering based Web Access Management approach for the prevention of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks.

Denial of Service and Distributed Denial of Service attacks mainly aim at the depletion of the victim's scarce system resources, which results in the denial of a network service to legitimate users whereas Distributed Denial of Service attacks make use of a whole series of poorly secured nodes, abusing them as handler/agents to amplify the attack.

Current defense approaches offer only very rudimentary means of defense: Some of them are over-simplistic and treat merely a small part of the problem by its symptoms, others only concentrate on the host's health state and do not care about the blocking of legitimate connections, yet other systems encounter serious difficulties of conclusion.

The main identified causes for these difficulties are information spoofing by attackers and the production of very high impacts over a short period of time, among others.

The present approach offers an alternative through the system "Fibered Guard", an intelligent personal firewall implementation, which successfully distinguishes legitimate connections from attack connections. This is done by empirical means, monitoring the system's health state and based on the benefits of Logical Fibering combined with a Data Mining conclusion engine.

A Logical Fibering is an abstract fiber space  $\xi = (E, \pi, B)$ , in which the fiber  $F$  over each point  $b$  in the base space  $B$  is a logical structure. "Fibered Guard" makes use of a fibered Fibering structure for apt data representation which includes the reduction of connection data to its primitives and the possibility of mounting logical structures directly upon the data. The connection data is stored in a Primary Data Representation, which contains raw information of incoming connections and a Secondary Data Representation, which represents rule sets to be applied for the blocking of attacks.

The Data Mining engine is used to support the process of data extraction from the Primary Data Representation - which might be compared to a pattern recognition process - and the transfer of results to the Secondary Data Representation. For this, a classification rule algorithm is used, which extracts rules for normal and problem system operation and mounts the respective structure in the Secondary Data Representation, which is afterwards employed for the intelligent identification of threats through a comparison module.

The Data Mining algorithm has been compared with an Artificial Neural Network (Multilayer Perceptron) as part of its validation.

In realistic simulations, Fibered Guard showed very successful against Denial of Service attacks, blocking imminent threats and keeping legitimate connections. Even in the more difficult Distributed Denial of Service attack environment, Fibered Guard offered very good effects against attacks and reasonable treatment of legitimate connections.

Future developments of this approach might lead to a powerful intelligent system for the generic defense against Denial of Service and Distributed Denial of Service attacks in home, commercial or industrial applications.

*Dedicated to my wife Alessandra*

# Acknowledgements

Firstly, I would like to thank Prof. Dr. Jacques Calmet for his excellent orientation and help any time I needed it. I have, indeed, been very lucky to have had such a good doctoral advisor! Yet again, it became clear that the University of Karlsruhe (TH) is an elite university with highly focused and open-minded professionals. I hope it will continue setting standards for German and international research.

I thank my second advisor Prof. Dr. Jochen Pfalzgraf for all his corrections and good advice. I am proud to have had such a direct contact with a brilliant researcher and professor.

Special acknowledgements also go to my fellow PhD candidates, especially Dr. Regine Endsuleit for her friendly and very precious help with organizational matters, which – as external candidate – were not at all simple for me in the first place. Dr. Yi Yang and Mr. Thilo Mie, thank you very much for all our fruitful discussions and your tips concerning the presentation!

Equally, Mrs. Helga Scherer, secretary of the Institute of Algorithms and Cognitive Systems has been a great support for all my organizational questions and remote problems. Thank you very much!

I could not write these acknowledgements without considering Prof. Dr. João Luís Garcia Rosa of the Pontifícia Universidade Católica de Campinas, who was the best master's advisor I could have, introducing me to solid scientific work and thus building the basis for this project.

I should still mention Prof. Dr. Alex Itiro Shimabukuru, also of the P.U.C. Campinas, who helped me to digest some tricky formulas and enriched my creativity.

Finally, but definitely not less importantly, I would like to thank my family:

I would like to stress how much I appreciated the active support and great patience of my family (Alessandra, Patrick and Katharina). Truly, without it, this project would not have gone far.

I would like to thank my parents, Prof. DDr. Hans Joachim Schneider and Hildegard Schneider for their motivating conversations. They gave me the basis to continue every day on this “non-trivial” journey.

# Contents

<b>1</b>	<b>INTRODUCTION.....</b>	<b>1</b>
1.1	THE INTERNET: A SAFE PLACE?.....	1
1.2	HISTORICAL REMARKS.....	1
1.2.1	Computer Generations.....	1
1.2.2	Roots of Programming Languages.....	3
1.2.3	Rollout of the Internet.....	4
1.3	SOCIAL ISSUES AND CYBERCRIME.....	6
1.4	CHALLENGES.....	7
1.5	JURISDICTION ISSUES.....	7
1.6	TECHNICAL SECURITY APPROACHES.....	8
1.7	INTRODUCTION TO DENIAL OF SERVICE ATTACKS.....	8
1.8	THESIS STRUCTURE.....	9
<b>2</b>	<b>STATE OF THE ART.....</b>	<b>10</b>
2.1	DENIAL OF SERVICE (DOS) AND DISTRIBUTED DENIAL OF SERVICE (DDOS) ATTACKS.....	10
2.1.1	Main Attack Types.....	10
2.1.2	Taxonomy.....	12
2.1.3	Selected Examples.....	13
2.1.4	Defense Methods.....	15
2.2	LOGICAL FIBERING.....	23
2.2.1	Formal Definitions of Underlying Principles.....	23
2.2.2	Logical Fiberings and Polycontextural Logic.....	24
2.2.3	Decomposition.....	25
2.2.4	Bivariate Operations.....	26
2.2.5	Tranjunctions.....	27
2.2.6	Types of Logical Fiberings.....	27
2.2.7	Applications.....	28
2.3	WEB ACCESS MANAGEMENT (WAM).....	29
2.3.1	Identity and Access Managment (IAM).....	29
2.3.2	Internet Access Management.....	29
2.3.3	Firewall Architectures.....	31
2.3.4	Tunnels.....	33
2.3.5	Applications.....	33
2.3.6	Scientific Approaches.....	35
2.3.7	Defense against DoS.....	36
2.4	DATA MINING.....	39
2.4.1	Introduction.....	39
2.4.2	Definition.....	39
2.4.3	Data Mining Process.....	39
2.4.4	Main Tasks.....	42
2.4.5	Supporting Technology.....	43
2.4.6	Techniques of Data Mining.....	43
2.4.7	Typical Applications.....	45
2.4.8	Market Analysis.....	46
2.4.9	Future Challenges.....	46
<b>3</b>	<b>A LOGICAL FIBERING BASED APPROACH.....</b>	<b>48</b>
3.1	INTRODUCTION.....	48
3.2	MOTIVATION.....	49
3.3	MAIN ALGORITHM.....	50
3.4	MODULE OVERVIEW.....	51
3.5	DESCRIPTION BY SYSTEM MODULES.....	52
3.5.1	Primary Data Representation.....	52

3.5.2	<i>Data Mining Conclusion Engine</i> .....	54
3.5.3	<i>Secondary Data Representation</i> .....	58
3.5.4	<i>System Monitoring Engine</i> .....	60
3.6	THEORETICAL JUSTIFICATION .....	61
3.6.1	<i>General Considerations</i> .....	61
3.6.2	<i>Pattern Recognition in Logical Fibers</i> .....	62
3.6.3	<i>Treatment of Typical DoS Attack Challenges</i> .....	64
3.7	COMPARISON AND VALIDATION .....	65
3.7.1	<i>Use of Artificial Neural Networks</i> .....	65
3.7.2	<i>Comparison with Approaches in Literature</i> .....	67
3.8	IMPLEMENTATION .....	69
3.8.1	<i>Main Directives</i> .....	69
3.8.2	<i>Feasibility Issues</i> .....	69
3.8.3	<i>Overview</i> .....	72
3.8.4	<i>Technical Details</i> .....	73
3.8.5	<i>Dynamic Environment</i> .....	79
3.8.6	<i>Limitations</i> .....	80
<b>4</b>	<b>RESULTS</b> .....	<b>82</b>
4.1	OVERVIEW .....	82
4.1.1	<i>Motivation</i> .....	82
4.1.2	<i>General Method</i> .....	82
4.1.3	<i>Test Setup</i> .....	82
4.1.4	<i>Measures</i> .....	83
4.1.5	<i>System Parameters</i> .....	83
4.2	DEFENSE AGAINST ATTACKS .....	83
4.2.1	<i>DoS Attacks</i> .....	84
4.2.2	<i>DDoS Attacks</i> .....	89
4.3	COMPARISON WITH OTHER APPROACHES .....	91
4.3.1	<i>Straight Forward Approach</i> .....	91
4.3.2	<i>Congestion Control Algorithm</i> .....	92
4.4	VALIDATION WITH ARTIFICIAL NEURAL NETWORK .....	93
4.5	TUNING .....	96
4.5.1	<i>DM Analysis</i> .....	96
4.5.2	<i>Rule Precision</i> .....	96
4.5.3	<i>Rule Validity</i> .....	97
4.5.4	<i>Save Rate</i> .....	98
<b>5</b>	<b>CONCLUSION</b> .....	<b>99</b>
5.1	ACHIEVED BENEFITS .....	99
5.2	KNOWN LIMITATIONS .....	100
5.3	FUTURE DEVELOPMENTS .....	101
	<b>REFERENCES</b> .....	<b>103</b>
	<b>APPENDIX</b> .....	<b>112</b>
	APPENDIX A: INTERFACE DESCRIPTION .....	112
	APPENDIX B: DATABASE CONFIGURATION .....	120
	APPENDIX C: PARAMETERS .....	122
	APPENDIX D: CLASS AND METHOD DETAILS .....	122
	<b>CURRICULUM VITAE</b> .....	<b>152</b>
	<b>PUBLICATIONS</b> .....	<b>154</b>

# Abbreviations

ACC	Agent Coordination Context, Aggregate Based Congestion Control
ALH	Admission List Housekeeping
ANN	Artificial Neural Network
ARPANET	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CAPPS	Computer Assisted Passenger Prescreening System
CERN	European Organization for Nuclear Research
chargen	Character Generator
COBOL	Common Business Oriented Language
CPU	Central Processing Unit
CRI	Critical Resource Index
DDoS	Distributed Denial of Service
DFA	Deterministic Finite Automaton
DNF	Disjunctive Normal Form
DNS	Domain Name System
DoS	Denial of Service
FATF	Financial Action Task Force
FG	Fibered Guard (implementation)
FIFO	"First-in-First-out"
FL	Fuzzy Logic
Fortran	Formula Translation
FQ	Fair Queuing
FTP	File Transfer Protocol
GB	Gigabyte
HD	Hard Disc
HTML	Hypertext Markup Language
IAM	Identity and Access Management
IBC	Illegitimate Blocked Connections
IC	Illegitimate Connections
ICANN	Internet Corporation for Assigned Names and Numbers
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System(s)
IKE	Internet Key Exchange
IP	Internet Protocol
IPTV	Internet-Television
ISP	Internet Service Provider
J2EE	Java 2 Enterprise Edition
JAAS	Java Authentication and Authorization
KDD	Knowledge Discovery in Databases
L2TP	Layer two tunneling protocol
LBC	Legitimate Blocked Connections
LC	Legitimate Connections
LSI	Large Scale Integration
MARS	Multivariate Adaptive Regression Splines
MAS	Multiagent System
MIT	Massachusetts Institute of Technology
MLP	Multilayer Perceptron



OLAP	On-Line Analytical Processing
ORI	Overall Resource Index
OS	Operating System
PC	Personal Computer
PCL	Polycontextural Logic
PPTP	Point-to-Point Tunneling Protocol
RAFF	Remote Active Filtering Firewall
RAM	Random Access Memory
RBAC	Rule Based Access Control
RED	Random Early Detection
RPA	Remote Patron Authentication
SAM	Source Access Mining
SAR	Source Address Refining
SOAP	Simple Access Object Protocol
SOM	Self organizing map
SSO	Single Sign-On
SYN	Synchronize/start
SYN/ACK	Synchronize/acknowledge
TCP	Transmission Control Protocol
TDFA	Time-Dependent Deterministic Finite Automata
TFN	Tribe Flood Network
TIA	Terrorism Information Awareness
TPC-C	Transaction Processing Performance Council
TPS	Transactions per Second
UDP	User Datagram Protocol
UNIVAC	Universal Automatic Computer
VLSI	Very Large Scale Integration
VoIP	Voice over IP (IP phoning)
WAM	Web Access Management
WWW	World Wide Web

# 1 Introduction

## 1.1 The Internet: A Safe Place?

When in 1962, J.C.R. Licklider of MIT discussed a “Galactic Network” in a series of memos [LEI03], he surely was not aware that after 40 years his idea would have become more than reality, nor would he ever have thought about the current problems, which many Internet users face like viruses, worms, spam-mail, phishing, DoS and other extremely common threats on a network, which indeed experiences serious problems of control.

This may be partly true because of the very nature of community growth (similar problems are encountered in big cities, where crime rates used to be uncontrollably high and anonymity hides a single small offender [SCH87]), however, as a matter of fact the concrete architecture of the net was not projected to enforce security control on a central level. Actually, not even look-and-feel or limitations of development are enforced, which opens the Internet to rapid growth (and is certainly one of its main focuses of popularity) but also lets all participants act freely – perhaps far too freely, with anarchical results. Indeed, one might get the impression of entering a city after a battle of war or a village in the ancient “wild-west”:

- Destruction is omnipresent: Failures happen, sites are down, errors occur out of no obvious reasons;
- Plunderers lurk around: Viruses and worms appear “from nowhere” putting network nodes in danger throughout seconds, openly illegal sites and exchange services invite innocent (or not so innocent) users;
- Some groups offer protection: There is a series of approaches, many of which claim to be “the ultimate protection” for marketing reasons, whereas some of them are of dubious nature (and may themselves be considered the true security risk);
- Legal issues develop, however, slowly: Indeed, at the moment there is no instance, which could totally prevent crime on the net as there is no global consensus about the very definition of crime and law enforcement. For instance, online gambling might be illegal in some places, but in others, it is legal – and sites move to these places, getting out of reach of any legal action. Secondly, it might be used for secondary criminal activities as money laundering [COA06];
- There is constant fear: Many users avoid online banking applications or e-commerce sites, because they simply do not know (and cannot possibly figure out), if data is cloned and misused afterwards, or not. Quite undifferentiated fears spread [STU04].
- Secret services are present: Personal data may easily be mined and imprudent organizations can potentially inflict great damage on individuals. Organized crime starts to focus more and more on online activities [WIL02].

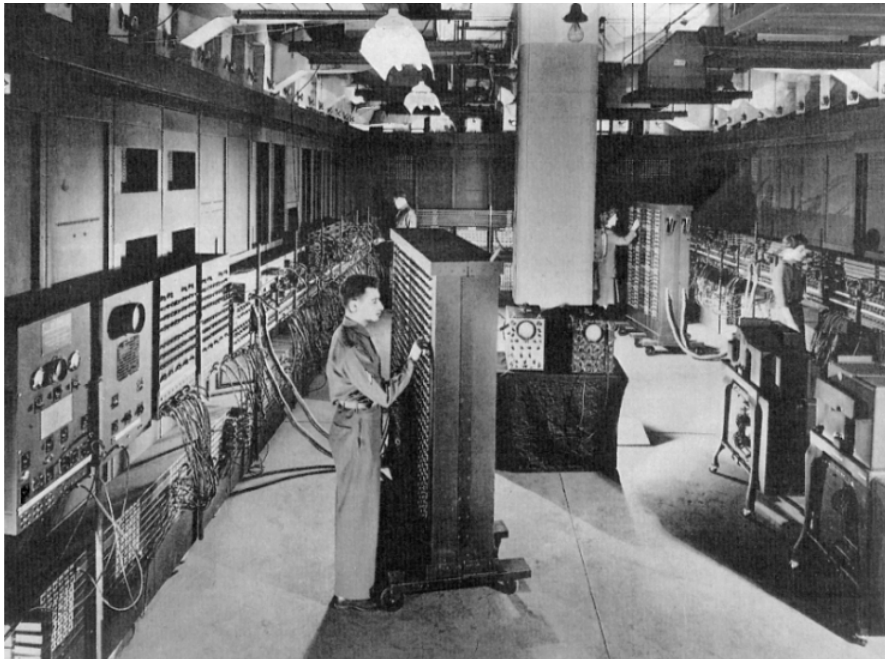
Truly, the Internet is not a safe place. However, its usefulness is out of question: In an infamous citation [W3C01] affirms that “The World Wide Web (known as “WWW”, “Web” or “W3”) is the universe of network-accessible information, the embodiment of human knowledge”. Facilities of communication, commerce and scientific exchange are developing and good solutions to the pressing security problems have to be found.

## 1.2 Historical Remarks

### 1.2.1 Computer Generations

There are different opinions of what should be considered the real first computer in history (see, for instance, [ZUS03]). Following [PAT98], we consider it to be the first electronic computer, namely the ENIAC (Electronic Numerical Calculator) (see Figure 1-1), which was constructed for

military purposes during World War II in the United States and made public only in 1947. It contained 18000 vacuum tubes, had enormous dimensions (80 feet long by 8,5 feet high) and could perform around 19000 additions per second. It also accepted conditional jump, which differentiated it from calculators. Programming was done manually by plugs and switches.



**Figure 1-1: “Inside” the ENIAC**

Whereas this computer was still held in the academic and military area, soon the first commercial electronic computers appeared and gave way to the development, which was the very origin of today’s computing.

According to [PAT98], the generations of computers, might be classified the following way (Table 1-1):

Generation	Dates	Technology	Principal new product
1	1950-1959	Vacuum tubes	Commercial electronic computer
2	1960-1968	Transistors	Cheaper computers
3	1969-1977	Integrated circuit	Minicomputer
4	1978-?	LSI and VLSI	Personal computer and workstations

**Table 1-1: Computer generations**

Since the beginning computer development was a vast area of research and there are several large publications on the subject (e.g. [CER03, IFR01]). This introduction shall only cite some important punctual dates in order to characterize the phases of development:

- The UNIVAC I (Universal Automatic Computer), which was one of the first commercial electronic computers, being sold for about \$1 million, had a total of 48 units. This first computer as many of the following was programmed by punched cards (similar to the one in Figure 1-1 [CRU04]), i.e. programming was slow, complex and tedious whereas the computers were by their very nature “stand-alone”. Security issues were not addressed, (neither thought of) or simply reduced to the access control of the room, where the computer could be found. There was a high amount of criticism toward the future development of computers as the “highly specialized machines” did not show very handy yet for general purposes [PAT98].

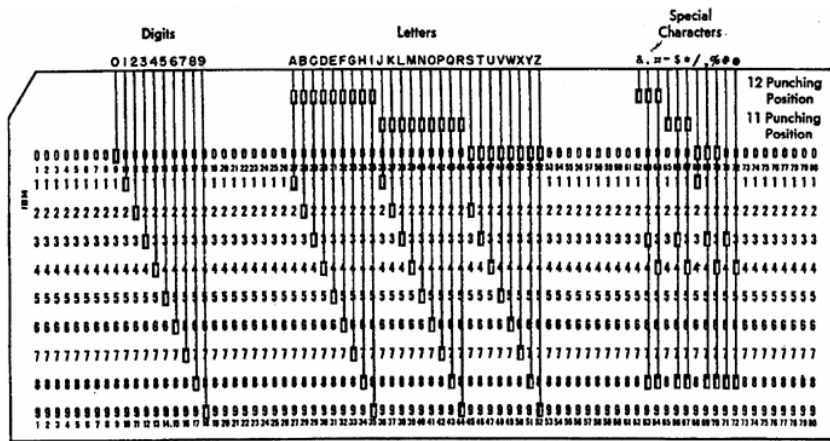


Figure 1-2: Example of a punched card layout

- Investing around \$5 billion, in the year 1964 IBM announced the System/360 (see [AMD64]), which should transform architecture abstraction to a commercial reality. Clock frequencies arrived at around 2 MHz and RAM memory at 256KB. The computer revolutionized the market by a much higher performance, which outnumbered the UNIVAC I by more than 250 times. In parallel the Digital Equipment Corporation launched the first Minicomputer: the PDP-8. Though yet expensive with a price of \$20000, it initiated the low-cost computer manufacturing, which was about to start. Yet another trend in a different direction over the same years produced the first supercomputer (CDC6600), designed by Seymour Cray, which was meant to be the best and fastest computer ever (with surely also the highest cost).
- One of the first computers for home use was the Apple-II designed by Steve Jobs and Steve Wozniak in 1977 [MOR06]. Whereas its functionality was highly limited [WEY07], it showed the way to the first IBM PC (see Figure 1-3) [SAB95], released in 1980/81 [VW07], which could be considered the first personal workstation. It initiated the popularization of computers, firstly in office and later also in home environments. With this distribution, broadly spread networking became possible.



Figure 1-3: IBM PC

## 1.2.2 Roots of Programming Languages

While [SAM72] shows that tracing back to the exact dates at which programming languages were launched, may be a really difficult task, the very first language might be considered the 1952 short code for UNIVAC. Basically, programming languages were developed in order to free

programmers of the highly error-prone task to enter with machine code, addressing directly registers.

The first higher language to be widely used was Fortran, developed around 1954. Its name was a shortening of Formula-Translator and its main focus was on mathematical facilities. Meanwhile a Fortran subroutine could not yet call itself and number storage easily caused programming errors [MIT02]. Fortran suffered a series of changes from its original version in the forthcoming years to keep it alive on the market.

In commercial ambiances and for business applications, Cobol was one of the most important early developments, with English language key words and machine independence [SAM72], still in use nowadays through the inertia of commercial applications [EKS03].

For academic purposes and mainly around Europe, Algol-58 was a further good initial development, which was improved in several versions. Its main innovation came 1970 with the implementation of record sets and abstract data types, which led the way to objects.

Finally LISP introduced procedural programming and was initially used for Artificial Intelligence applications, later to be substituted in popularity by Prolog. It was one of the first languages to develop stack management and implement recursion.

Language	Expressions	Functions	Heap storage	Exceptions	Modules	Objects	Threads
Lisp	x	x	x				
C	x	x	x				
Algol 60	x	x					
Algol 68	x	x	x				x
Pascal	x	x	x				
Modula-2	x	x	x		x		
Modula-3	x	x	x	x	x	x	
ML	x	x	x	x	x		
Simula	x	x	x			x	x
Smalltalk	x	x	x	x		x	x
C++	x	x	x	x	x	x	
Objective C	x	x	x			x	
Java	x	x	x	x	x	x	x

**Table 1-2: Comparison of programming language concepts**

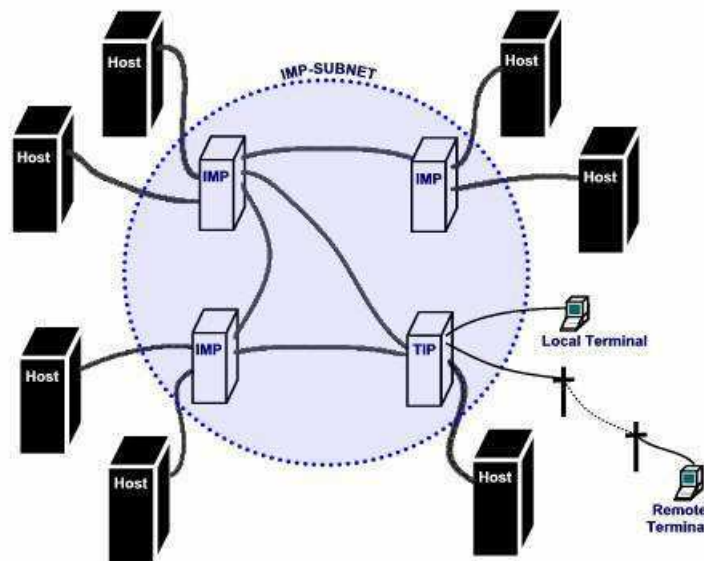
Comparing several programming languages (see Table 1-1), one might observe that some concepts are covered by a whole series of languages, whereas others are reserved to the more current paradigms. For networking purposes, thread treatment might be considered in particular important, as several events might happen at the same time and need to be successfully treated. It should be said that currently object-oriented languages are widely used (such as Java and C++). More on the choice of the programming language of the implementation conducted during this approach may be found in Chapter 4 (“A Logical Fiberling Approach”).

### 1.2.3 Rollout of the Internet

Many of the security problems encountered nowadays may be traced back to the origins of the Internet, which started as a small visionary academic project and the development of which was not at all foreseen in its creation days [LEI03]. There is a whole range of papers about the history of the Internet. For clarity reasons, we follow [ROB06], who divides the history of the Internet in four stages:

- Stage 1: Whereas the origins might be seen much before with the installation of the ARPANET (see Figure 1-4) [YUR07, LEI03], the Internet itself started to exist in the

1980s (during a phase when home users found that the first home computers came into shops worldwide). This first stage mainly focused on research and the academic environment. The development was sponsored by federal U.S. funding. In the beginning of the 1980s there was no consensus yet upon a common protocol to be used. A critical date in the scenario was January, 01, 1983 when TCP/IP was chosen to substitute NFS on the ARPANET. This may be seen as crucial for the Internet known today. Although TCP/IP developed in order to be usable by PCs, it still remained its initial structure, which was basically defined by simplicity and interoperability [ROB06]. Furthermore it was projected to be very scalable, supporting practically infinite growth of the number of network nodes (as a matter of fact they grew from around 100 hosts in 1985 to more than 500 million nowadays) and to be technically reliable (the U.S. military thought of it as a communication system, which should be able to withstand an atomic attack) [LEI03]. There was definitely no focus on data security in the early days. This is why the Internet's core architecture until today cannot be considered safe. Surely, several enhancements happened over the years, but they were executed on higher layers only.



**Figure 1-4: Possible ARPANET architecture, visualizing its IMP-subnet ring communication topology**

- Stage 2: Naming it the “Early Public Internet” (1992-1997) [ROB06] describes how the Internet was opened to the public interest by the creation of Internet Service Providers (ISPs) and the formation of the World-Wide-Web (WWW) on what was mainly conducted until then by simple file transfer and text interfaces, which were not likely to attract the attention of a large public (like those offered on the V100 terminal shown in Figure 1-5 [WIL01]). The WWW was based on a proposal of Tim Berners-Lee on the implementation of a more flexible document management system (initially called “Mesh”) at CERN (European Organization for Nuclear Research). It was to be a quick and simple system, based on Hypertext (coined in the 1950s by Ted Nelson). Main advantages appointed by [LEE90] were the remote access, support for heterogeneity, non-centralism and facilities of data analysis. Security was not an initial requirement. In the early years the Internet became steadily known by enterprises and government institution. Prominent support came 1993 from the Clinton government when the White House implemented the High Performance Computing act and deployed the Internet throughout governmental institutions. This sign of implicit approval was one of the important happening for the Internet's further development in domestic, commercial, industrial and governmental sectors. Yet in 1996 a new approach to Internetworking was launched by more than 30

universities, the “Internet-2”. At stage 2 several security issues were known and already pressing. Treatment policies and systems were implemented, but as the number of users was small and naturally more specialist users had access, control still remained easier.



**Figure 1-5: VT100 terminal for text-only access to the Internet**

- Stage 3: During the third stage, which [ROB06] dates from 1998 to 2005, the Internet reached high domestic popularity. Internet stocks went up on the stock exchange until 2000 and then collapsed as they were merely based on speculation. Broadband Internet became popular and fiber optics spread at low prices after 2000. The technical administration of the Internet was shifted from research agencies to the Internet Corporation for Assigned Names and Numbers (ICANN). Security issues became more and more imminent, especially because of the generally poor security of home network nodes. This was one of the reasons why Denial-of-Service (DoS) and especially Distributed-Denial-of-Service (DDoS) attacks could develop, whereas common defense methods did not show very efficient on this new form of attack (more on this in Chapter 2).
- Stage 4: In the fourth stage, which includes the present and the future, the Internet will have to face challenges of becoming truly the most important means of communication, by further implementing IP-phoning (VoIP), Internet-Television (IPTV) and other very demanding services [ROB06]. It may be asked, whether security issues should be addressed in a conventional high-level way, whether they should be more supported by legal action or should be treated at the very core (which would mean a global roll-out of a new technology).

### **1.3 Social Issues and Cybercrime**

The Internet may be seen as a mirror of the human society, with all its advantages and problems [ROB06]. This can be traced back to the fact that it is a means of fast communication and knowledge propagation. As it reaches high numbers of users, common human mentalities become more and more the focus and some of them could be compared to the Roman satire of Juvenal (“Panem et Circenses”), which basically meant that people wanted to be sustained and entertained, even on very low levels in the infamous Roman circus. This is one of the main problems any media faces. Some drastic (and perhaps anecdotic) approaches therefore see the Internet as a way to the apocalypse [HOR04].

Meanwhile, in the relatively open environment, a feeling of anonymity helps users to commit acts on the Internet they would actually not consider if their deeds were publicly known. In [GRA08] some explanations are given concerning the nature and reasons of Cybercrime, which can be traced back to the following set of conditions:

- **Motivated offenders:** Actually, the medium is new, but many offenses are simply inserted into a new environment. Human feelings as greed, lust, revenge, and adventure spirits promote most of them. As the only difference [GRA08] mentions yet the challenge to dominate complex systems, which might be attributed, for instance, to a great universe of hackers. It is clear that the pool of potential offenders rises with the number of growing users on the Internet.
- **Suitable targets:** There is a whole series of possible victims from e-commerce sites to innocent users, who are misled and abused by others [GRA08]. Furthermore, means of communication is greatly amplified, which enables individuals to coordinate criminal action and even inexperienced users (“script kiddies”) to launch powerful attacks.
- **Absence of capable guardians:** Guardians could be technical (defense systems), legal (law enforcement on the Internet) or yet social (parental control) [GRA08]. Though there are even prominent cases of successful treatment, there are yet more prominent cases when defense mechanisms failed and services were put to the mercy of criminals. Especially, DoS and DDoS attacks have had a great potential of inflicting damage on well-known victims. More on this in Chapter 2.

Legal treatment of the new global medium of communication is still very far from feasible, as legislation is not yet a global issue [GRA08]. Criminal activity can concentrate in weak states, information safe havens, which makes true global action necessary [WIL02], i.e., the “gaps” must be all closed. The response must thus be strategic, multi-level, multilateral and transnational [WIL02].

Furthermore, it shall be pointed out that apart from isolated criminal activities, the Organized Crime, too, has discovered the Internet as a means to propagate activities. In this case the network is used as a tool and therefore the aim – unlike with hackers – is not its disruption, but its exploitation [WIL02]. Money laundering, fraud and theft, which are typical crimes committed, often lead to a criminal organization rather than individuals. There is a strong rising trend for this form of crime [WIL02].

## 1.4 Challenges

In fact, there is a series of specific challenges any action – if by technical, legal or social means – faces [GRA08, WIL02]:

- **Tractability:** It is difficult to identify the source of an attack and – if identified – to clearly define a person behind it. Evidences may be easily falsified in a way to put the very term “evidence” in question. Sources of evidence, moreover, can be easily deleted.
- **Constant change:** The Internet changes constantly, technically and socially. These changes reflect in user behavior and must be treated. Traditionally, defense systems deal with this issue by constant updates, even if this can never offer really “full protection”. Laws are meanwhile a little reluctant to follow the constant change, needing prompt action to adjust to the ever changing environment.
- **Velocity:** With the advent of new Internet technologies, it may be guessed that attacks will gain even greater computational power and put performance issues at the very front of a defense approach development.
- **No central enforcement:** The Internet, by its core architecture, is a federated structure. There is no way of centrally controlling and enforcing policies.

## 1.5 Jurisdiction Issues

In order to treat the problem of global action, which is clearly identified by lawyers and researchers of the area, several approaches have been done to harmonize state laws. One of them was the Financial Action Task Force (FATF), set up by the G-7, which launched a “name and shame” campaign, indicating 15 jurisdictions, where combat to the problem of money



laundering (earlier mentioned) was completely inadequate [WIL01]. The results were remarkable changes in these jurisdictions, and the effort may give an idea of how global action could be initiated in the future.

It should be mentioned that furthermore many countries throughout the globe that noticed the problematic development the Internet would provide through its anarchic character, created national laws to offer protection in their jurisdiction [DRE06]. This is, e.g., true for the United States and the European Union. Main focuses are domains and names, copyrights and patents, contracts and data security. Law suits based on these norms may have civil or criminal consequences, however, in practice, mainly in a determined country.

Another interesting step into the correct direction may be observed by the institution of the International Association for Artificial Intelligence and Law (IAAIL) in the early 1990s [IAA07], which sponsors the International Conference on Artificial Intelligence and Law (ICAAIL) events, currently in their eleventh edition. Although the focus is generally Artificial Intelligence applied to law or utilized for legal issues, there is also the opposite focus of legal action in the cyber world as shown by the tutorial [ZEL01].

## 1.6 Technical Security Approaches

There is a whole range of commonly used security systems. Some popular security systems on client computers are [MAR07, CER07]:

- **Antivirus:** Antivirus systems treat viruses, worms and Trojans and might also be extended to other security problem treatments. They typically maintain a large local virus data base, which has to be constantly updated, scan loaded programs and e-mails.
- **Personal Firewall:** In order to prevent unauthorized access to a client machine, personal firewalls overview the incoming and outgoing connections and block anything that may appear suspicious. Typically, they are not very intelligent and prompt the user, which leads to potential misconfigurations.
- **Anti Spyware:** Spywares are mainly cookies, which are downloaded and “spy out” the behavior of a user on the web, in most cases for advertising purposes. An Anti-Spyware system scans cookies and other critical system areas for this kind of intruder.
- **Anti Malware:** Malware is a set of different malicious software products. Anti Malware scanners may treat viruses, Trojans, spyware etc. all in one.
- **Verification of Running Processes:** Often Antiviruses fail to correctly examine and alert the user of malicious running processes. Therefore, separate utilities exist to perform this task; however, they normally presuppose an advanced user.
- **Anti Keyloggers:** A Keylogger is a system, which logs keystrokes on a client machine and afterwards transmits them to the intruder. This way mainly passwords but also other classified information may be spied. Anti Keyloggers examine keyboard hooks and alert users of the fact that “somebody might be listening”.
- **Anti DoS systems:** Yet very sparsely implemented, Anti DoS systems treat the specific problem of Denial-of-Service attacks. Examples are: Collapsar AntiDoS System [NSF06] or FloodGuard Alert [FIS03]. The field is truly quite unexplored (compared to other security areas) and needs intelligent action, as the one proposed in this thesis.

## 1.7 Introduction to Denial of Service Attacks

This thesis presents an approach for the defense against Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. These attacks aim at the disruption of services on the Internet and are accomplished by the attacker sending interfering traffic to the victim [MIR04]. This interfering traffic in a typical case causes an overload situation on a scarce system resource (like bandwidth, open requests, memory etc.). It might, however, also aim at a system or architecture

bug. The real characteristic for identifying DoS attacks, thus, is their outcome (service disruption) and not their method [MIR04].

In this work, we understand DDoS attacks as a sub-group of DoS attacks. They are characterized through the fact that an attacker recruits a series of poorly secured nodes on the net in order to amplify an attack, which seeks an overload situation [CER01].

Basically any node on the Internet can be a victim, but there is a main trend to attack e-commerce sites, large providers or even root servers of the Internet [MCC03]. This makes the attacks financially important. There may be yet other critical consequences, as shown by the activity breakdown at the port of Houston, which was provoked, by DDoS attack activity [MIR04].

Defense systems have great difficulties to avoid DoS attacks as they present mainly two heavy challenges [MIR04]:

- They are hard to identify because they use to disguise themselves as legitimate traffic;
- They may be launched in high volume, which could drown any activity of dividing traffic into “legitimate” and “illegitimate” by the roots.

There is a series of interesting approaches, which, however, concentrate on only a small part of the picture parting from imprecise assumptions, or do not take sufficient care to not fall into the trap of being themselves abused for DoS attacks. Therefore, a current practice is moreover the use of a catalogue of general measures for prevention than the use of specialized systems [BMF02, DFN01].

A thorough introduction to DoS and DDoS attacks and more details on defense approaches may be found in Chapter 2 (“State of the Art”).

## 1.8 Thesis Structure

This thesis proposes an intelligent firewall, which makes use of the principles of Logical Fibering for apt data representation and a Data Mining engine for the establishment of conclusions.

The following part of the thesis is structured in this way:

- Chapter 2 (“State of the Art”) gives an idea of the current research concerning DoS/DDoS, Logical Fibering, Web Access Management and Data Mining;
- Chapter 3 (“A Logical Fibering Based Approach”) introduces the concrete defense approach, its principles, justifications and implementation details;
- Chapter 4 (“Results”) provides proof for the functionality of the approach giving detailed results and comparisons;
- Chapter 5 (“Conclusion”) gives a list of benefits and mentions future perspectives and applications.

This thesis makes use of several figures to facilitate the understanding of its course and principles. Whenever these figures were obtained or adapted from existing publications, they are cited together with their references throughout the main text.

## 2 State of the Art

This work is aiming at designing a new methodology and at implementing a system to prevent Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. The present chapter outlines the state of the art in this domain (see 2.1). Then, the theoretical foundation is based on Logical Fibering (Section 2.2). Web Access Management is a mechanism, which is mandatory for the use of the resulting system, which is why section 2.3 is devoted to its principles. Finally, Data Mining is selected as the tool to identify attackers. An overview of the relevant techniques is presented in the last section (2.4).

### 2.1 Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks

#### 2.1.1 Main Attack Types

Denial-of-Service attacks are attacks that aim at denying a specific service on the network to legitimate users. This may be done by flooding the network, thus substantially diminishing or blocking legal traffic, by disrupting connections, by preventing certain individuals from access or by blocking a specific service or system.

Attacks are typically launched via software, but may also include a physical destruction of equipment in order to reach the same effect. Surely, the Denial-of-Service attack may be part of a larger attack, being an intended or even a not intended consequence [CER01, LAU00].

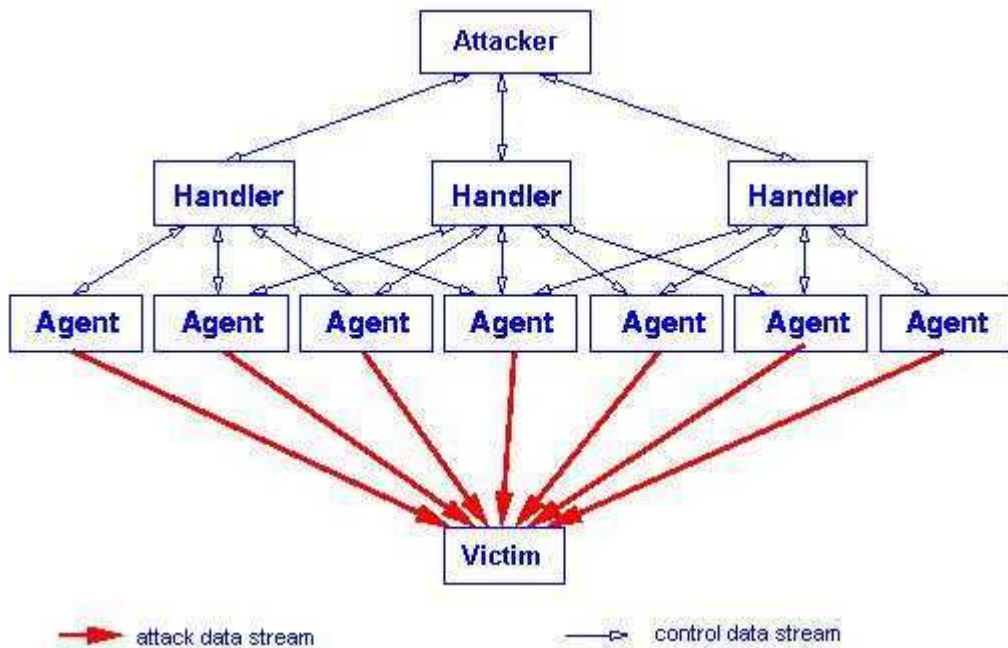
Denial-of-Service attacks may completely disable servers, clients or an entire network. In case of organizations that are doing business over the Internet, the effect may be a disaster, in fact disabling the organization itself and causing loss of thousands of dollars in a short period of time [CER01, MCC03].

Basically, Denial-of-Service attacks are simple to launch and difficult to prevent. Users with rather limited resources may attack a large sophisticated site ("asymmetric attack") [CER01].

There is already a number of tools which employ a user-friendly interface and considerably simplify action for beginners. Furthermore it is not required to brake into the victim's system itself as in most cases widely available services are attacked [MCC03].

Victim administrators, on the other hand, might not interpret the situation correctly as an attack, but could see it as a simple lack of resource or instant error, thus often only rebooting the machines or amplifying resources and not taking the appropriate action [MCC03, ULB03].

Distributed Denial-of-Service (DDoS) attacks are an evolution and subgroup of Denial-of-Service attacks. In this form of attack the intruder does not use only a single machine to attack a system directly, but makes use of a whole network of nodes, which greatly amplifies the power of the attack. Figure 2-1 (see below) shows a typical architecture of a DDoS attack. The attacker breaks into vulnerable systems, installs handlers and agents in a clandestine manner and then launches an attack remotely [DFN01, ULB03].



**Figure 2-1: DDoS attack architecture**

To achieve this, the handlers are contacted via control data streams, which then instruct the agents to attack the victim. Thus, the victim merely has contact with the agents, which brings about difficulties of tracing, makes the attack more fault-resistant and – depending on number and power of the agents – can even bring large sites down as was the case with Yahoo!, eBay, Buy.com, CNN.com and others [MCC03].

According to [SHI02] another subdivision should still be made, establishing a class of Network Denial of Service (NDoS) attacks. [SHI02] gives the following definition: “A *network denial-of-service attack* occurs when some set of network entities intentionally uses network services with the goal and effect of causing consumption or corruption of network resources in such a way that some other set of network entities have their ability to access otherwise usable network services degraded or so delayed as to render them unusable.” NDoS attacks – this way – are a subgroup of DoS attacks, with the special characteristic of involving only network entities.

Basically the following modes of attack exist:

- Consumption of scarce resources: In the most typical scenario an attacker will aim at the consumption of a scarce resource, thus producing an overload situation with the given impacts. This may be done preventing network connectivity by producing a high number of faulty connections (see the “SYN flood attack” below). Another way would be the consumption of bandwidth (e.g. “ICMP ECHO attack”), the creation of processes to consume CPU time, the explosion of disc-spaces or the misuse of password policies to prevent legitimate users from access. Instabilities could be caused by the sending of unexpected data and the attacker might even use the victim’s resources against itself by making it send attack data over the network for him [CER01].
- Alteration of configurations: A service may malfunction or not function at all if configurations are altered. This applies to computers, but also especially to routers, where a change might bring down a whole network [CER01].

- Physical attacks: Attackers might physically destroy elements of the victim's network or break into server rooms, thus simplifying the attack greatly for the intruder and reaching the desired effect even with powerful security policies in place [CER01].

## 2.1.2 Taxonomy

In order to get a complete overview of attack and defense methods in the ambience of DoS, [MIR04] proposes two quite complete and relatively complex models, which are depicted in Figure 2-2 and Figure 2-3 below.

According to [MIR04], DDoS attacks shall be classified mainly by the following characteristics:

- Degree of automation: Is it a manual, semi-automatic or automatic attack? In the case of semi-automatic attacks several subgroups are given (see Figure 2-2).
- Exploited weakness of deny service: Is it an attack taking advantage of a bug, or is it a brute-force attack via flooding?
- Source address validity: Is the source address valid or spoofed and – if spoofed (as in most cases) – what technique was used and how can the information be obtained?
- Attack rate dynamics: Is there a constant or variable attack rate?
- Possibility of characterization: May the attack be characterized by the header fields of the packet?
- Victim type: Who has been attacked?
- Impact on the victim: What were the consequences of the attack?

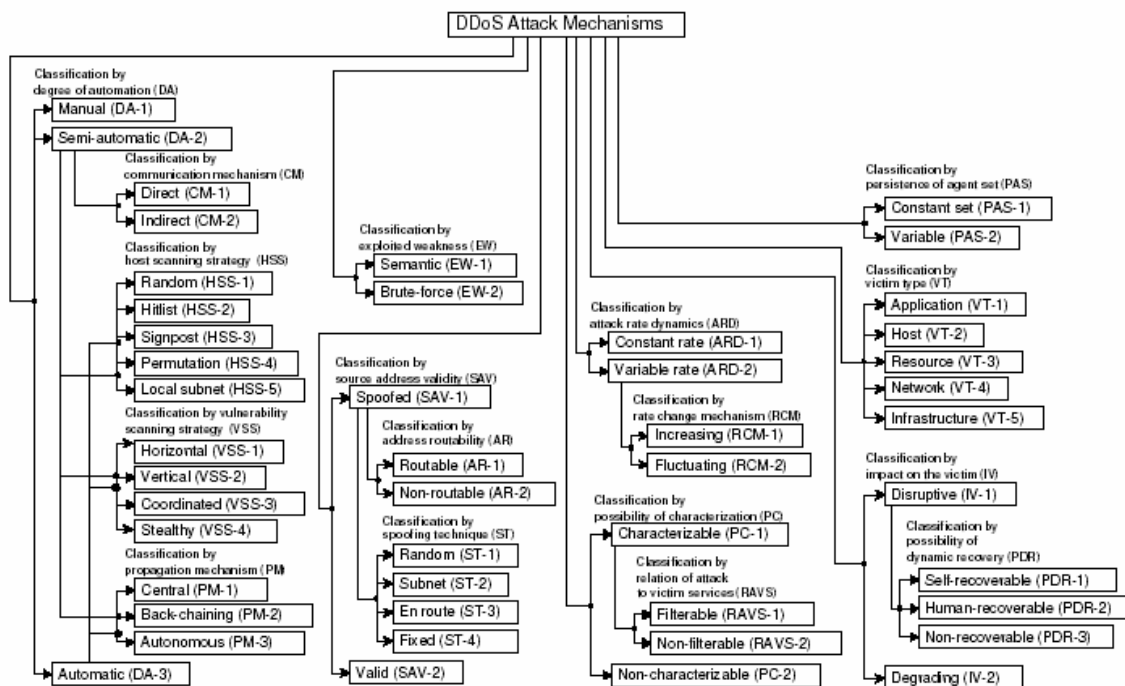
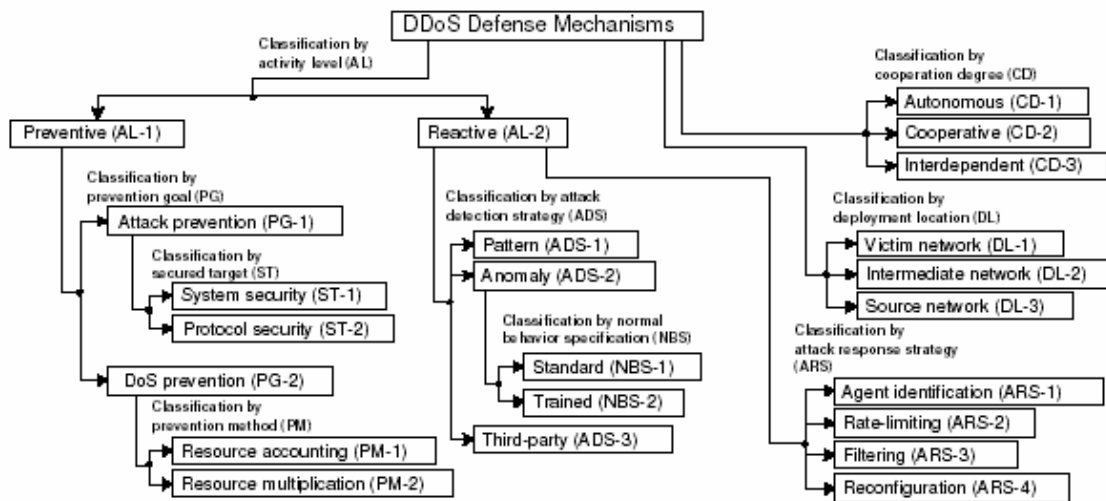


Figure 2-2: Taxonomy of DDoS attack mechanisms

In the same way [MIR04] classifies the defense mechanisms. Here are the characteristics that are crucial (compare Figure 2-3):

- Activity level: Is the defense preventive (and in what form) or reactive (and how)?
- Cooperation degree: Is the defense system stand-alone or do several systems take part?
- Deployment location: Where is the defense deployed inside the attack environment?



**Figure 2-3: Taxonomy of DDoS defense mechanisms**

It is important to observe that this is one possible form of taxonomy and that very different opinions about the way of classification exist (e.g. [HUS03]).

### 2.1.3 Selected Examples

There is a large domain of DoS attack methods, which is constantly growing due to the development of the existing techniques as well as the detection of new bugs, which can be taken advantage of. This section is concentrating on examples of the most well-known tools and methods in order to convey an idea of the spectrum of DoS-attacks.

The simplicity of a DoS-attack, which allows even beginners to cause huge problems, may be studied through the “Smurf” attack. This attack uses a large amount of ICMP echo traffic, which is directed to one or several IP broadcast addresses in networks that answer to such a requests, thus creating a flood of useless traffic [LAU00].

First the attacker sends ICMP ECHO packets to the amplifying network’s broadcast address using a forged address so that it seems that actually the victim’s system started the request. Consequently all systems on the network, will answer to the fake victim’s request. Depending on the number of members of the network the victim will be completely flooded with answers and the traffic might make the network break down [LAU00, MCC03].

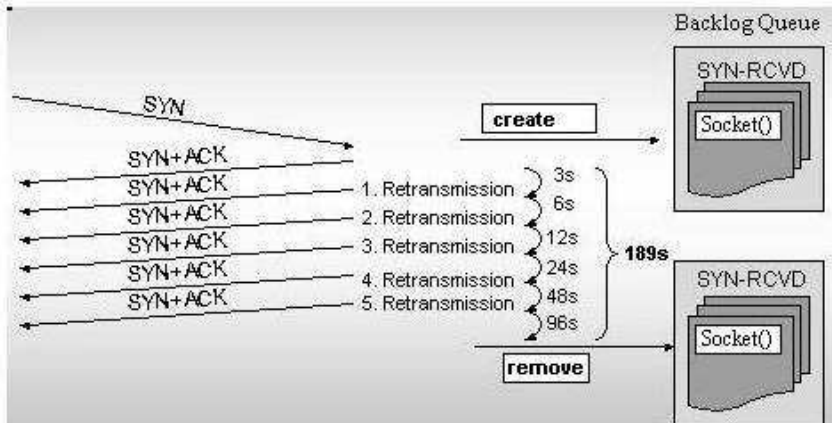
This form of attack also uses packets UDP (in this case called “Fraggle”).

Using the “Smurf” attack, the attacker not only uses the victim’s capacities against itself, but also greatly amplifies the impact, what might be still only a tiny problem if just one attacking machine was involved. This is a principle, which may be observed in many different DoS-attacks.

Another problem consists in the design and ingenious principles of some systems, especially concerning the frequently used TCP/IP-protocol, which was developed in the first place to ensure US military communication after a nuclear attack (see [BAR64]). Its main purpose was fault-tolerant operation in case a point of the network broke down. The problem of abuse, however, was not considered in the days of development as only few well-behaved users were taking part in the network. This is surely not true anymore in times of a global Internet (see Chapter 1). Popular attack forms as the SYN-Flood attack developed, taking advantage of this error concerning judgment of future possibilities.

This attack is based on exploiting the TCP three-way handshake using incomplete TCP-connections to make the victim arrive at the maximum number of connections, thus denying service to any one that requests connection [LAU00].

First an intruder sends a SYN packet from his system (using a forged, unreachable address) to the victim's system. The victim's system then answers SYN/ACK to the fake address. As the address is not reachable, the victim will never receive any answer and the requested connection will remain in the buffer queue at wait state (SYN\_RECV), only being terminated after the timeout period (see Figure 2-4 [SCH05]). If many SYN-requests like these are sent to the victim, soon the buffer will fill up and it will not be able to respond to new TCP-connection requests for a considerable time. Depending on the configured timeout each connection slot will be freed again. This, however, may take even more than 20 minutes for some systems [MCC03, ULB03].



**Figure 2-4: SYN-flood attack mechanism**

Another example of an architectural inadequacy being used to deploy a DoS attack is the UDP-Flood attack. In this attack the intruder uses forged UDP-packets to connect the echo-service on one victim machine to the character generator ("chargen") service on another machine. The packets sent from the machines to each other completely flood the connection between the two machines using all of its bandwidth [LAU00].

Ironically, DoS attacks may even use security systems against themselves, as shown in [JIN05]: In this case the abuse of trust management systems is studied. As they normally work by receiving and analyzing credentials from clients, these might well be tricked by sending very extensive credential strings and having the server analyze them (thus easily breaking down because of overload). Though there might be simple defenses (as shown in [JIN05]), these should have been considered already in the phase of architecture.

On the other hand known system bugs are misused to produce incorrect results, malfunction or complete breakdown. For example, DNS-Attacks pass erratic data to the Domain Name Server, forcing it to put wrong information in its cache (see [CHU05]). Thus, the name of a site may be mapped to an inexistent address, which has the effect of actually denying the service of the original site to all users of the Domain Name Server [MCC03]. This attack is possible with older BIND versions and needs the DNS recursion to be switched on [MCC03].

Another example is the Teardrop attack, which takes advantage of a Linux kernel bug. When IP-packets travel through the network, it might be necessary to divide them in smaller parts, depending on the Maximum Transmission Unit. Older Linux kernels (or Windows versions) - though checking if a fragment is too large - never check whether it is too small. Thus, artificially produced faulty packets can bring down the machine instantly [DFN01, MCC03].

Finally, several tools were developed and launched in order to facilitate DDoS-attacks. They employ the typical DDoS architecture consisting, generally speaking, of an attacker that breaks into several machines with weak security, which then serve to attack the victim. This way, it is actually not necessary to break the victim's security. The tools discussed here are: TFN, TFN2K, Trinoo, WinTrinoo and Stacheldraht.

Being the first publicly known DDoS tool, TFN (Tribe Flood Network) makes use of a client-server structure, installing the server on a remote system and afterwards remotely activating a large-

scale DoS attack including Smurf, SYN-Flood, UDP-Flood and ICMP-flood attacks [DTT99, LAU00].

TFN2K is an advanced version of TFN, which uses random ports for communication and encryption based on the CAST-256 algorithm [LAU00, MCC03].

Trinoo works in a similar way as TFN, also making use of a client-server structure, this time instructing a master (handler) to command the daemons (agents) to attack.

WinTrinoo is Trinoo for Windows, which works through the installation of a Trojan horse (normally in service.exe). When installed, it writes a line into the registry in order to execute every time the system is started up [MCC03]. One of the most developed tools for DDoS is Stacheldraht (ger. "barbed wire"). A complete analysis of this tool may be found in [DIT99]. Stacheldraht is a tool that combines the resources of TFN and Trinoo. It offers the possibility to run an encrypted telnet session between client and master using a symmetric key [MCC03, ULB03]. Also, an automatic update technique for the attack daemons is used [LAU00]. User commands are given using a command shell.

## 2.1.4 Defense Methods

Several methods of defense against DoS-attacks were developed. This section is to give an overview and offer a basis for later discussion and contrast to this thesis' approach.

### 2.1.4.1 Catalogue of Recommendations

There are some general recommendations, which use to be provided to users and administrators, in order to cope with Denial of Service Attacks. These recommendations are normally quite simple and intend to eliminate the gross part of the attacks and/or smoothen the effects of attacks, which actually get through nonetheless, i.e., it is clearly stated that further attacks are still possible. Some examples of these recommendations are [BMF02, LAU00, CER01, DFN01]:

- Implementation of router filters in order to prevent fake addresses ("IP Spoofing");
- Installation of all available patches for the system, thus treating known security problems;
- Disabling unused services to expose the system to threats only up to the necessary extent;
- Conservative password policies: users shall not be given abusive rights;
- System quotas to prevent disc space explosions;
- Monitoring of the system and establishment of limits for normal operation in order to not ignore circumstances of an attack;
- Redundant and fault tolerant network configurations;
- Use of authentication and encoding schemes;
- Use of tools to detect configuration changes;
- Turning off of directed broadcasts;
- Use of Open Source products, as bug fixes and patches are developed quicker for them;
- Regular scanning for DDoS components (with Find\_DDoS, RID, Zombie Zapper and other specific utilities);
- Use of hot backups for important machines, in case the normal ones become completely inoperable;
- Implementation of backup policies to save all important data regularly;
- Regular security trainings with the staff so they become aware of imminent threats.

### 2.1.4.2 Basic Approaches

This section shall give an idea of basic approaches, which do not use sophisticated techniques, but yet produce interesting results.



- Ingress/Egress-Filtering: Being a classic approach, Ingress and Egress-Filtering as shown in [FER98] and [SAN00] is based on the fact that a legal request from a specific domain must provide an IP-address from that domain – any other IP address would be fake and therefore indicate a Denial-of-Service attack. Ingress-Filtering is done at the Internet Service Provider where all packets with illegal source address (on the base of their ingress link) entering the network are filtered away. Egress-Filtering takes place at the exit point of the customer network, filtering all messages with addresses that are found illegal [SAN00]. The major downside of this approach is performance, because every packet has to be checked. This needs extra hardware and implies in costs, which leads ISPs normally to not implement this filtering technique. Furthermore, this approach does not actually eliminate the possibility of IP spoofing, but reduces the possible universe to all legal addresses in a specific domain [TUP03].
- Client puzzles: The interesting approach of client puzzles in [AUR00] suggests the use of puzzles for the authentication of clients that intend to connect to servers, thus freeing the server from time-consuming extra work which may be needed to authenticate the client by other means (e.g. by a SYN-cookie, i.e. a message that is sent by the server to the client and must be returned in the same way to authenticate). The formula used is shown in Figure 2-5. The transmission scheme may be described as follows: As soon as the level of open connections becomes problematic, the server switches from normal operation to puzzle authentication. In this operation it periodically adjusts the difficulty level ( $k$ ), generates the server's nonce ( $N_s$ ) and sends a signed and time stamped broadcast to the client. The client verifies the message, generates its own nonce ( $N_c$ ) and the brute-force solution to the puzzle ( $X$ ) and sends a signed message with this data as well as the received server nonce ( $N_s$ ) back to the server. The server then verifies that  $N_s$  is recent and that the solution is correct as well as the signature of the client. With everything correctly given, the client is authenticated and informed about this result. Communication may begin.

$$h(C, N_s, N_c, X) = \overbrace{000 \dots 000}^{\text{the } k \text{ first bits of the hash}} \underbrace{Y}_{\text{the rest of the hash bits}}$$

$h$	= a cryptographic hash function (e.g. MD5 or SHA)
$C$	= the client identity
$N_s$	= the server's nonce
$N_c$	= the client's nonce
$X$	= the solution of the puzzle
$k$	= the puzzle difficulty level
$000 \dots 000$	= the $k$ first bits of the hash value; must be zero
$Y$	= the rest of the hash value; may be anything

**Figure 2-5: Formula for client puzzles**

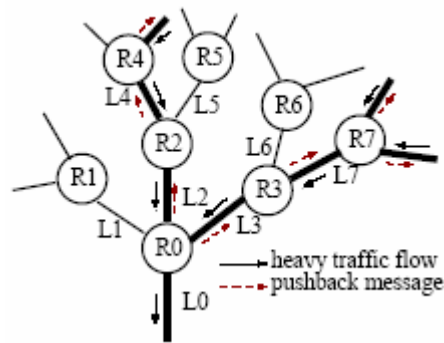
### 2.1.4.3 Tracking Algorithms

One of the main problems of DoS-attacks is the difficulty to track them as normally spoofed information is provided in order to hide the real attacker's identity from the eyes of the victim. Tracking algorithms shall help to improve this situation by gathering information of the attack at different sources and thus trying to arrive at conclusions concerning its path and origin.

- Center Track: In his paper [STO00] Robert Stone gives an approach that uses an overlay network. The implementation shall be done using adjacent tracking units at all edge routers with a central control router. Possible attacks are traced from the router nearest to

the victim up to the entry point of the network, thus reducing the necessary steps to trace all the attack back. This method is called hop-by-hop tracking. CenterTrack also uses a second ingredient, which is the so called traffic flow measurement [STO00]. In this method, record is maintained of all TCP connections including the connections that presented attacks. These records are kept at every router and connections that brought about DoS-attacks are readily discarded from their detection onwards. Thus, it is an empirical system, which might be problematic because of a considerable overhead of router activity as well as incorrect blockings that might be based on false record data.

- IP Traceback: [SAV00] presents algorithms that use marking of packets at routers in order to reestablish the path at the victim's machine. Incrementing the data at every router surely represents a considerable overhead of information; however, also more economic approaches exist where not every router inserts information into the packet, but only some routers might do so randomly. Meanwhile it is possible to carry a value for the hops in a separate variable, which remains annexed to the packets. The path might be constructed also with only a few markings, as attacks use to create a high amount of the same network traffic.
- Tracing of spoofed packets: Another trace back technique [BUR00] traces a route from the victim that is under attack to every network, using network mapping technologies. It follows the assumption that a short burst on the path which the attack stream uses, will actually alter the intensity of the attack, partly blocking it. This way, beginning with the closest router, bursts are sent to the network and when the attack stream is altered, the connections from the specific destination router are tested until the way of the attack is found out by this "trial and error" scheme.
- Aggregate Based Congestion Control (ACC): Proposed by [MAH01] ACC consists of two algorithms: local aggregate congestion control and router pushback. The main idea of [MAH01] to identify attacks on the network is to discover aggregates, i.e. segments with similarities, which might be addresses in the packets, length, utility etc. including especially characteristics that could mark attacks like TCP SYN packets or ICMP ECHO packets. All in all, the scope may be defined freely, i.e. very widely or very narrowly – depending on the purpose of the analysis. Local aggregate based congestion control is made of an identification algorithm, which shall identify all congestion causing aggregates as well as a control algorithm, which shall limit the bandwidth for this form of aggregate and avoid an imminent breakdown. The purpose of pushback is to limit aggregate traffic already at adjacent upstream routers forcing them to reserve certain rates for the traffic, like this diminishing or eliminating completely the attack streams from the requesting router to the victim. The pushback message may go recursively to all routers that receive heavy traffic flow (as shown in Figure 2-6). The process shall start with the nearest routers and continue to the outer routers. The main problem in this case is to determine if congestion is due to chronic overload or an attack and secondly to find a reasonable rate to reduce harmful traffic – these are problems that [MAH01] also mentions. Surely, empirical measures that are constantly adjusted may be adequate in this case, as each situation is different.



**Figure 2-6: Pushback scheme**

#### 2.1.4.4 Network Congestion Control

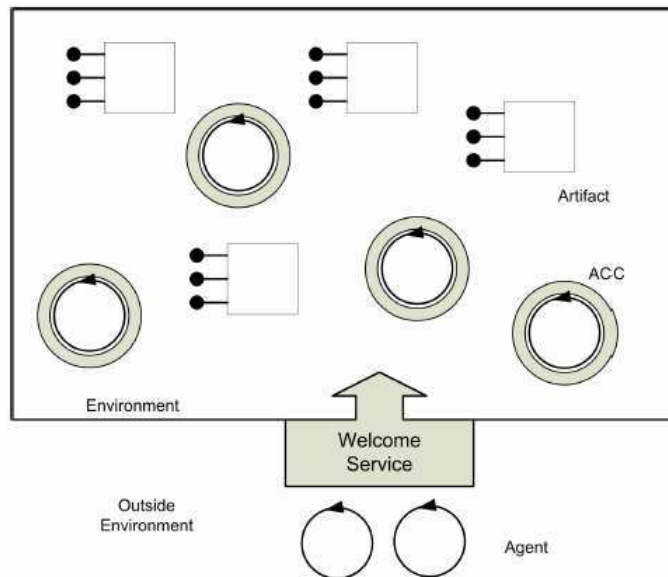
Several approaches see DoS as a problem that could be handled by congestion control, done by scheduling and queue management at the routers. Some examples are:

- Fair Queuing (FQ): The input traffic is divided into several queues and each queue receives an equal share of bandwidth. Thus, one queue cannot degrade the qualities of another, but the implementation is costly as divisions have to be made. [DEM90] Efforts have been made in order to mitigate the undesired overhead, developing variations of this algorithm (e.g. in [STO98] & [MCK90]), though without fully satisfactory success.
- Random Early Detection (RED): A very simple system to get rid of congestions is RED [FLO93]. In this system of queuing, all requests enter into a simple FIFO and as soon as congestion problems come up, the router randomly drops requests. Though this surely solves the router's problem, it does not at all solve the problem of DoS as it prevents legitimate users from usage by not distinguishing the DoS traffic flows. One of the several variations of RED is given in [FLO99]. In this paper this "extreme unfairness" is being dealt with as well as the problem of potential congestion collapse. To treat the problem, [FLO99] suggests router mechanisms that identify and restrict the bandwidth of selected high-bandwidth best effort flows in times of congestion. These flows may be unresponsive, "not TCP-friendly" or using disproportionate bandwidth.
- Differentiated Services: [BLA98] defines an architecture for implementing scalable service differentiation on the Internet. This architecture achieves scalability by aggregating traffic classification. Packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path. Given this framework the necessary classifications, policing and shaping may easily be implemented at the boundaries of the network or on hosts. Using these services, a large variety of anti-congestion environments can be established, i.e., a high flexibility is achieved at the cost of possible configuration errors.
- On-Off Feedback Control: Proposed by [XIO01] the On-Off Feedback Control uses a backward pressure propagation feedback control scheme for defense from DDoS attacks. Algorithms for rate-based and queue-length based feedback control were developed using two different performance metrics. The system works with threshold values that – once exceeded – trigger a feedback response in order to adjust bandwidth. [XIO01] is based on a simple practical example of implementation on a small network. This scheme may also show efficient, but does not differentiate between legitimate and DoS traffic.

#### 2.1.4.5 Intelligent and Advanced Approaches

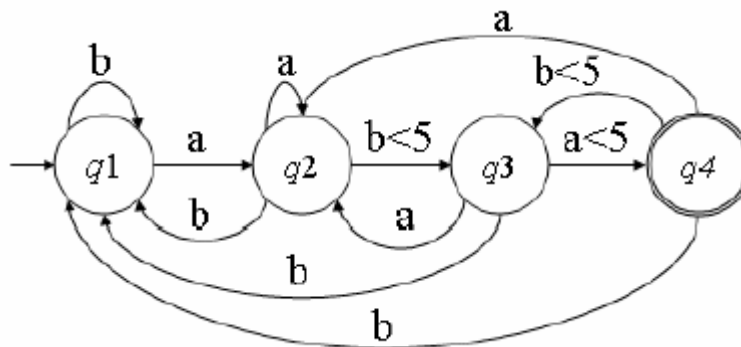
Intelligent approaches are those that use methods of Artificial Intelligence to obtain their results, namely listed here: Data Mining, Intelligent Agents, Automata Structure Heuristics and Artificial Neural Networks.

- A Data Mining Approach to Intrusion Detection: Lee and Stolfo discuss in their interesting approach [LEE98] the use of Data Mining facilities in order to detect intrusions on the basis of selected characteristics. This is done via classifiers that may classify situations into “normal” and “abnormal” depending on previously learned data, as well as link or sequence analysis in order to establish relations between certain phenomena. The system is maintained flexible so that the administrator may change the characteristics, which are being analyzed and deploy new versions of the logic as needed. [LEE98] makes use of RIPPER, a rule learning program, to learn from the training data. Whereas [LEE98] determines that even classifiers depending on single characteristics may be effective, they also offer a second idea of a combined classifier, which might be more powerful and accurate concerning results. It may be criticized that this approach is actually a straight forward Data Mining approach and the results may depend a lot on the data, which is obtained. Though Data Mining can be a good way of actually finding rules behind attack schemes, it bares the risk, of producing a reasonable set of uncertain conclusions, which have to be validated through other forms of analysis.
- Mobile Agent Systems: [JAN00] mentions a number of reasons for the use of mobile agents in IDS (intrusion detection systems), actually arriving at the conclusion that they are the ideal way of processing in this scenario. [JAN00] defines a software agent as “*a program that can exercise an individual’s or organization’s authority, work autonomously towards a goal, and meet and interact with other agents.*” meaning that emphasis is put on the characteristics of autonomy and social ability. To run mobile agent defense systems, widely available agent platforms must be available. Advantages given for the application of mobile agents are [JAN00]:
  - Overcoming of Network Latency: The agents’ mobility allows them to analyze network nodes directly and take action where normal systems might not reach.
  - Reduction of the Network Load: Agents may move directly to the data for analysis. Thus, network traffic is considerably reduced.
  - Autonomous and Asynchronous Execution: Mobile Agents survive problems happening at the creation site and may work completely on their own.
  - Dynamic Adaptation: Being one of the main agent characteristics, the possibility to react to changes may be vital in unknown attack scenarios.
  - Platform Independence: Agents use to be high-level code in order to insure their mobility over OS-boundaries.
  - Protocol Encapsulation: The communication protocol may be directly incorporated by agents, allowing them to launch interface upgrades when moving to other hosts.
- An application of a concrete MAS for intrusion detection is given in [GAR06]: The approach tries to model a very interesting scenario, in which network intrusions are compared to virus/bacteria intrusions into the human body. The MAS tries to mimic the natural defense mechanism: It implements static barriers by an Agent Coordination Context (ACC), which defines methods that can be used by agents that try to access the resources on the network. It is the central instance to provide a RBAC (Rule Based Access Control). This way of authentication and authorization is completed by a dynamic environment, using agents called “agentLy”, which shall implement the task of lymphocyte cells in a human body, i.e., try to identify misbehaving agents and warning the ACC of violations (see Figure 2-7). Each agent has the following characteristics:
  - Diversity: Only a subset of bad behavior is analyzed,
  - Disposability: If no attack is detected over reasonable type, the agent may be recycled,
  - Adaptability: New abnormal behavior shall be learned.



**Figure 2-7: Architecture of the proposed MAS imitating the human immune system**

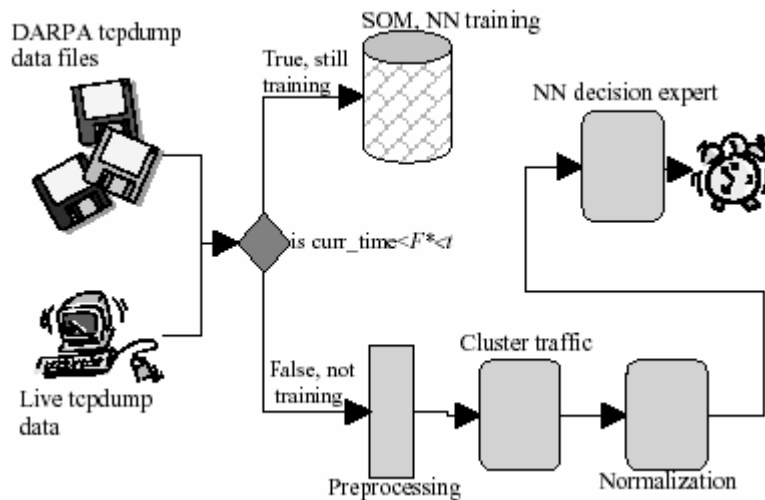
- Automata: [BRA02] develops an approach to use Time Dependent Deterministic Finite Automata (TDFA) for intrusion detection - specifically for the detection of DoS attacks (see Figure 2-8 for an example TDFA [BRA02]). This approach is based on the concept that DoS attacks follow certain attack sequences in a timely fashion, which may be detected by applying a form of finite automata with state transitions that do not only consider changing variables but also the time when they change. [BRA02] demands that there is something as a "language of DoS" and that therefore similar methods might be used as for compilers of programming languages. The system itself consists firstly of preprocessing units for noise filtering such as the data filtration unit and the token generator and secondly of the core logic in the TDFA transversal unit and the TDFA provider. As special advantages of the system [BRA02] mentions the fact that it may work the same way on real time and historical data and that it supports updates during operation. According to the results, which are presented, the system is efficient against several well known forms of attack as SYN Flood, Ping Flood, Teardrop and Smurf.



**Figure 2-8: Example of a TDFA**

- Neural Networks for Intrusion Detection: [BIV02] presents a connectionist approach to intrusion protection. This is done via the careful analysis of network data. The system employs a classifying self-organizing map (SOM) for data clustering and an MLP (Multilayer Perceptron) neural network for detection. The data is obtained by information provided through the *tcpdump* command. In the first phase - the architectural learning

period - appropriate data is learned in order to establish the basis for intrusion analysis. SOM and MLP are trained at the same time according to a training scheme shown in Figure 2-9 (see [BIV02]). In productive operation, data is first clustered by the SOM and then passed on to the MLP for analysis. Criticism to this algorithm may be firstly that neural networks tend to be “black boxes” not allowing insights to their conclusions, thus keeping users dependent on the machine for any analysis. Secondly, the MLP architecture may not be ideal as training sets are being “forgotten” soon. This might be the reason why [BIV02] mentions that his network had troubles to detect all forms of attack at once.



**Figure 2-9: Operation scheme of the Neural Network for Intrusion Detection**

- **DDoS Defense by Offense:** This interesting approach given in [WAL06] shall protect servers from application Denial of Service attacks. It assumes that attackers use a very high bandwidth on the network, which goes until the very limit of their capacities, whereas good clients still have possible space. In a possible attack situation the server should, instead of blocking traffic, encourage all clients to “speak up”, i.e. multiply their traffic. The expected result would be that the good clients “crowd out” the bad ones, thus defending effectively against DDoS (see Figure 2-10). Though interesting and innovative by the idea, there are some severe critics on this approach: Firstly some conditions must apply (as given in [WAL06]) for the approach to work in the expected way: There has to be an adequate link bandwidth and an adequate client bandwidth. This can – in general - not be expected. Secondly, the authors do not understand the real idea of DDoS: It does not at all technically matter if traffic is good or bad intentioned. The only figure that creates a successful attack is the traffic itself. Therefore it is truly counterproductive and noxious to multiply client traffic. The process of “crowding out” seems problematic. It might be that bad clients this way cannot get the influence they intend; however, as the overall situation gets crowded, this does not really matter. This way, even though the approach is an interesting reminder that unconventional thinking might bring a yet better solution to the problem, it still has to be matured.

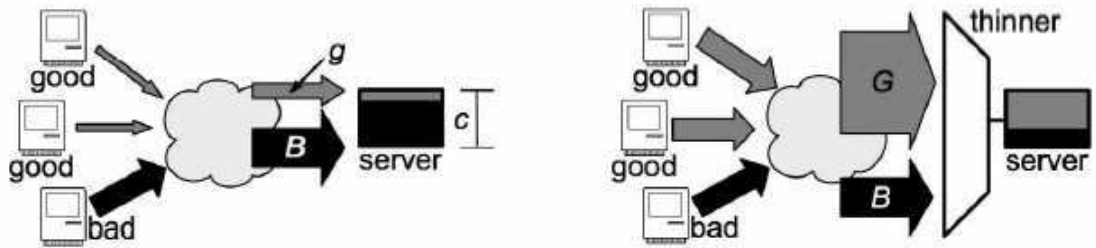


Figure 2-10: Attack situation without “speak up” (left) and with “speak up” (right)

## 2.2 Logical Fibering

The following part shall give an overview of the principle of Logical Fibering. The present approach uses Logical Fibering non-trivial data representation of the DoS attack situation and exploits the main advantages of Logical Fibering.

### 2.2.1 Formal Definitions of Underlying Principles

#### 2.2.1.1 Vector Spaces

Let  $F$  be a field, the elements of which are called scalars. A vector space over the field  $F$  is a set  $V$  with two binary operations [WEI03]:

- Vector addition:  $V \times V \rightarrow V$ , denoted  $v + w$ , where  $v, w \in V$ ,
- Scalar multiplication:  $F \times V \rightarrow V$ , denoted  $av$ , where  $a \in F$  and  $v \in V$ .

These operations shall satisfy the following axioms:

- Vector addition:
  - For all  $u, v, w \in V$ , we have  $u + (v + w) = (u + v) + w$ ,
  - For all  $v, w \in V$ , we have  $v + w = w + v$ ,
  - There is a zero vector  $0 \in V$ , such that  $v + 0 = v$ , for all  $v \in V$ ,
  - For all  $v \in V$ , there is an element  $w \in V$ , called *additive inverse* of  $v$ , such that  $v + w = 0$ .
- Scalar multiplication:
  - For all  $a \in F$  and  $v, w \in V$ , we have  $a(v + w) = av + aw$ ,
  - For all  $a, b \in F$  and  $v \in V$ , we have  $(a + b)v = av + bv$ ,
  - For all  $a, b \in F$  and  $v \in V$ , we have  $a(bv) = (ab)v$ ,
  - For all  $v \in V$ , we have  $1v = v$ , where 1 denotes the multiplicative identity of  $F$ .

#### 2.2.1.2 Fiber Bundles

A fiber bundle consists of the data  $(E, B, \pi, F)$ , where  $E, B$  and  $F$  are topological spaces and  $\pi: E \rightarrow B$  is a continuous surjection.  $B$  is the base space of the bundle,  $E$  the total space,  $F$  the fiber and  $\pi$  is the projection map [ROL99].

We require that for any  $x$  in  $B$ , there is an open neighborhood  $U$  of  $x$  (which will be called a trivializing neighborhood) such that  $\pi^{-1}(U)$  is homeomorphic to the product space  $U \times F$ , in such a way that  $\pi$  is the projection onto the first factor. A fiber bundle could be expressed by the following diagram:

$$\begin{array}{ccc} \pi^{-1}(U) & \xrightarrow{\phi} & U \times F \\ \pi \downarrow & & \text{proj1} \downarrow \\ & & U \end{array}$$

$\text{proj1}: U \times F \rightarrow U$  is the natural projection and  $\phi: \pi^{-1}(U) \rightarrow U \times F$  is a homeomorphism. The set of all  $\{(U_i, \phi_i)\}$  is called a local trivialization of the bundle.



For any  $x$  in  $B$ , the preimage  $\pi^{-1}(x)$  is homeomorphic to  $F$  and is called the fiber over  $x$ . A fiber bundle  $(E, B, \pi, F)$  is often denoted  $F \rightarrow E \xrightarrow{\pi} B$ .

In another definition [STE51], a fiber bundle may also be seen as a coordinate bundle with all possible coordinate functions of an equivalence class.

A vector bundle is a fiber bundle, where the typical fiber  $F$  is a vector space [ROW99].

### 2.2.2 Logical Fiberings and Polycontextural Logic

The concept of Logical Fibering was developed by J. Pfalzgraf ([PFA91]) during a project based on Polycontextural Logic (PCL), involving two university groups as well as an industrial company where he was system engineer.

Logical Fiberings have shown to be especially adequate for the use in Multiagent systems (see [PFA00] and [PFA01]).

Polycontextural Logic (PCL) was introduced by G. Günther based on his work on philosophy and cybernetics [GÜN62]. Later work on PCL was done by R. Kaehr and coworkers.

One basic idea is the viewpoint that the world consists of a distribution of two-valued logics ('*loci*') at different ontological places. These logical places are held together by a connection, which is addressed by G. Günther as "a mysterious function called self-reference". Thus the universe is understood as a "unit of contextual existence and coexistence", meaning that the whole system is more than just a collection of its elements.

The following shall give a formalization of these two-valued logics and their self-reference. It shall be noted that PCL can be interpreted as a special form of Logical Fibering (explained below) and that one of its main principles is the decomposition of a many-valued logic into classical two valued logics.

A logical space in a PCL system is denoted by  $L_i, i=1,2,\dots,n$ , where  $n = \binom{m}{2}$  and  $m$  is the number of the global truth values, thus the PCL system can be interpreted as an  $m$ -valued logic with  $n$  classical subsystems.

From a global point of view the total system may be seen as a co-product of the two-valued logics, following  $L^{(m)} = L_1 \coprod \dots \coprod L_n$  and the truth values of the local systems are in a certain relation, expressed by a so-called mediation scheme. Figure 2-11 show the mediation scheme of the simplest case of a PCL system ( $m = 3$  and  $n = 3$ ).

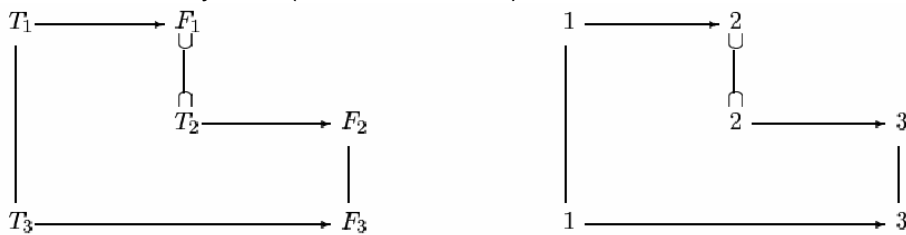


Figure 2-11: Example of a mediation scheme

The arrow  $T_i \rightarrow F_i$  expresses an ordering in the subsystem  $L_i$ . The symbol  $\supset \text{---} \subset$  expresses a change from a  $T$ -value to an  $F$ -value and vice versa (here from the  $F$ -value in  $L_1$  to the  $T$ -value in system  $L_2$  and vice versa). The vertical arrows express identification of the truth-values. This mediation scheme can be interpreted as the following equivalence relation on the set of all local truth-values:

$$T_1 \equiv T_3; F_1 \equiv T_2; F_2 \equiv F_3$$

Denoting the equivalence class of  $T_i$  by  $[T_i]$  these are the three global values (see right side of Figure 2-11):  $1 = [T_1] = [T_3]$ ,  $2 = [F_1] = [T_2]$ ,  $3 = [F_2] = [F_3]$ .

Certainly, the scheme for  $m = 3$  is rather simple and with increasing numbers of local subsystems more complex relations arise, c.f. the following mediation scheme for  $n = 6$  and  $m = 4$  [PFA01]:

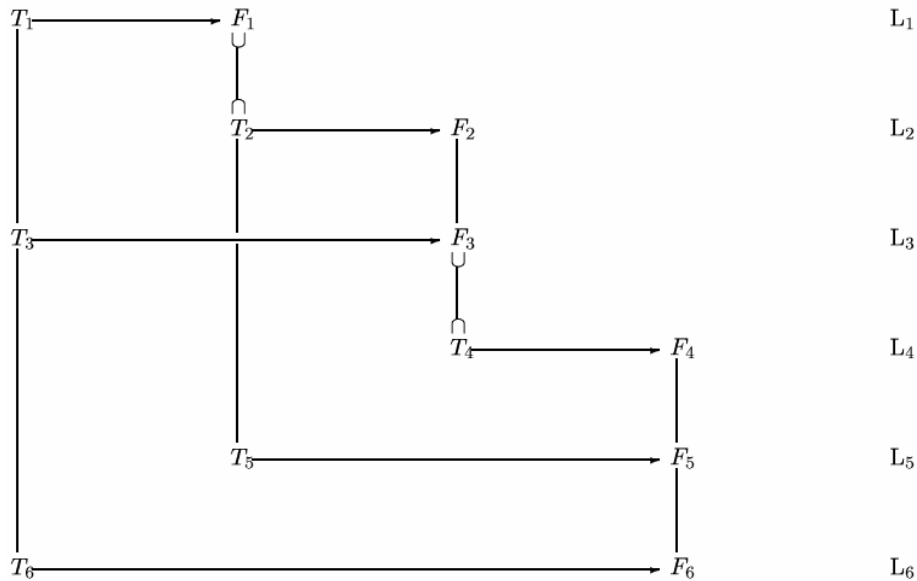


Figure 2-12: Example of a mediation scheme (MS4)

The mediation scheme in Figure 2-12 corresponds to this  $\equiv$ -relation:

$$T_1 \equiv T_3 \equiv T_6; F_1 \equiv T_2 \equiv T_5;$$

$$F_2 \equiv F_3 \equiv T_4; F_4 \equiv F_5 \equiv F_6$$

### 2.2.3 Decomposition

A method for the decomposition of many-valued logics is given in [PFA95] - including an algorithm. The following shall give a short introduction.

Taking a truth table of a three valued logic (values  $T, *, F$ ), a conjunction (“AND” operation) may look the following way:

$\wedge$	$T$	$*$	$F$
$T$	$T$	$*$	$F$
$*$	$*$	$*$	$F$
$F$	$F$	$F$	$F$

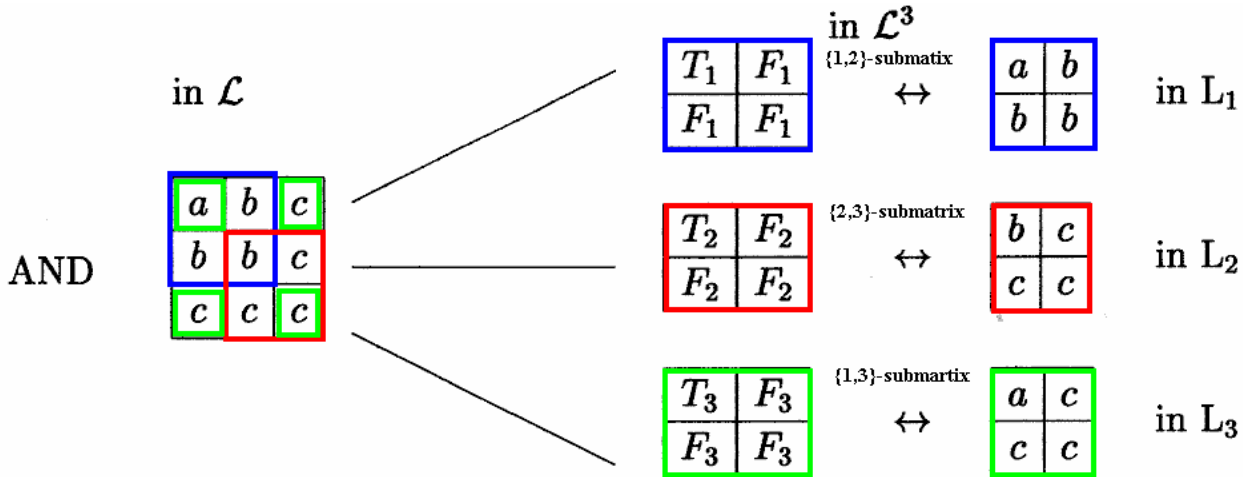
Table 2-1: Truth Table (“AND” operation on 3-valued logic)

For a simpler notation, column and row captions are suppressed and the truth values are renamed the following way:  $a := T, b := *, c := F$ , yielding the following shorter notation for the operation in Table 2-1:

a	b	c
b	b	c
c	c	c

**Table 2-2: Shorter notation of Table 2-1**

The global set of truth values, hence, is  $\Omega = \{a, b, c\}$ . This global logic shall be decomposed to the three classical two valued logics  $L_i, i = 1..3$  (with local values  $\{T_i, F_i\}$ ). The total universe of available local values is  $\Omega^3 = \{T_1, T_2, T_3, F_1, F_2, F_3\}$ . The decomposition is done by finding suitable equivalence relations on  $\Omega^3$  such that the set of residue classes  $\Omega := \Omega^3 / \equiv$  yields the global truth value set  $\Omega = \{a, b, c\}$ . The respective logical connectives (the “AND” operation in the current case) shall be implemented by local logical connectives in the subsystems.



**Figure 2-13: Decomposition method (“AND” operation on 3-valued logic)**

The decomposition method is illustrated by Figure 2-13 (adapted from [PFA95]). Firstly, from the truth table in  $L$  we derive three sub matrixes (defined by their limits): the  $\{1,2\}$ -sub matrix, the  $\{2,3\}$ -sub matrix and the  $\{1,3\}$ -sub matrix, considering that the  $\{i, j\}$ -sub matrix consists of the elements  $\{(i, i), (i, j), (j, i), (j, j)\}$ . In the above case it may be observed that each sub matrix gives the values of a classical conjunction truth table, which may be expressed by  $x_i \wedge y_i$  respectively. The present relation may furthermore be modeled by the mediation scheme in Figure 2-11 as the encountered relation follows  $T_1 \equiv T_3; F_1 \equiv T_2; F_2 \equiv F_3$  (see above). Finally, it may be expressed by the bivariate operation  $X \wedge \wedge \wedge Y$  [PFA95].

### 2.2.4 Bivariate Operations

Via tableau method, bivariate operations are introduced in PCL. With bivariate operations logical functions inside the subsystems may be modeled. For a system with three global truth values ( $L^{(3)}$ ) consequently  $X \wedge \vee \wedge Y$  means a conjunction in subsystem  $L_1$ , a disjunction in  $L_2$  and a conjunction in  $L_3$  respectively. These operations may also be expressed in the following vector notation:

$$X \wedge \vee \wedge Y = \begin{pmatrix} x_1 \wedge y_1 \\ x_2 \vee y_2 \\ x_3 \wedge y_3 \end{pmatrix}$$

It should be noted that not all such operations can be formed consistently (an example is  $X \wedge \vee \rightarrow Y$  in  $L^{(3)}$ ) [PFA91]. However, the concept of “transjunctions” may be used to solve this problem of incompatibility as demonstrated in [PFA95] for the case  $X \rightarrow \rightarrow \rightarrow Y$ . This shall be further explained in the following part.

### 2.2.5 Tranjunctions

Modeling the local bivariate operation as a map  $(\Theta : L_i \times L_i \rightarrow E)$  a distribution of the input pairs  $(x_i, y_i) \in L_i \times L_i$  over the maximally four subsystems  $(L_\alpha, L_\beta, L_\gamma, L_\delta)$  is possible (see Figure 2-14) [PFA96].

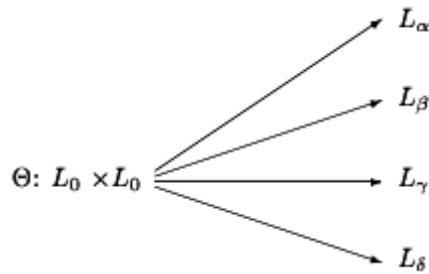


Figure 2-14: Transjunction

Considering the truth values, there are obviously four possible input pairs  $(\Omega_i \times \Omega_i = \{(T_i, T_i), (T_i, F_i), (F_i, T_i), (F_i, F_i)\})$ . Taken a classic truth-value matrix of a disjunction, we obtain:  $\Theta(T_i, T_i) = T_\alpha$ ,  $\Theta(T_i, F_i) = T_\beta$ ,  $\Theta(F_i, T_i) = T_\gamma$ ,  $\Theta(F_i, F_i) = F_\delta$ . This way, the transjunction is used to spread images over subsystems (see [PFA04]).

It may be applied in cases of inconsistencies produced by the decomposition process demonstrated earlier. E.g., in the case of the representation of an IMPLY operation (mentioned earlier), the decomposition process yields the following identifications on the value set  $\Omega^3$  (for details on the process see [PFA95]):  $a = [T_1] = [T_2] = [T_3]$ ,  $b = [F_1]$ ,  $c = [F_2] = [F_3]$ . The  $\equiv$ -relation  $T_2 \equiv T_1$ , which is produced (among others) is an inconsistency (unlike the  $\equiv$ -relation  $T_2 \equiv F_1$  in the cases of “AND” and “OR”). To solve this problem a suitable transjunction  $x_2 \Rightarrow_i y_2$  may be used, defined by the bivariate operation  $\Omega_2 \times \Omega_2 \rightarrow \Omega_1 \cup \Omega_2$  in  $L_2$ , distributing values over the subsystems  $L_1$  and  $L_2$ , corresponding to  $T_2 \mapsto T_1$ ,  $F_2 \mapsto F_2$ . The IMPLY operation can thus be expressed by  $X \rightarrow \Rightarrow_i Y$ , which is compatible with the original equation [PFA95].

### 2.2.6 Types of Logical Fiberings

A fiber space is a quadruple  $\xi = (E, \pi, B, F)$ .  $B$  is the base space,  $E$  is called total space,  $F$  is a typical fiber,  $\pi$  defines the projection map  $\pi : E \rightarrow B$ . An abstract fiber space is denoted by  $\xi = (E, \pi, B)$ . Following this, a Logical Fibering is an abstract fiber space, in which the fiber  $F$  over each point  $b$  in the base space is a logical structure (i.e. in most common cases  $F = L$ ,  $L$  being a classic two-valued logic).

A classic simple case of a Logical Fibering is a so-called free parallel system (trivial fibering). In this case  $\xi = (E, \pi, I, L)$ , i.e., the base space  $B$  is an index set  $I$  and the typical fiber  $F$  is a

classical first order logic space  $L$ . The total space is defined by  $E = I \times L$  and the fiber over the point  $i \in I$  is  $\pi^{-1}(i) = \{i\} \times L$ , also denoted by  $L_i$ . The respective truth values in  $L_i$  are denoted  $\Omega_i = \{T_i, F_i\}$ . As there is no relation between the local fibers (thus the term “free parallel system”), the global set of truth values is a co-product of the local ones:

$$\Omega^I = \coprod_{i \in I} \Omega_i$$

Cited as “Derived Logical Fiberings” in [PFA04], a Logical Fiberings with a nontrivial equivalence relation  $\equiv$  (as used with PCL systems, see above) on the set of all truth values ( $\Omega_I = \{T_i, F_j\}$ ) is denoted by  $L^{(I)}$ . In this case the set of global truth values is the quotient space following  $\Omega^{(I)} := \Omega^I / \equiv$ . A PCL system can be represented as a special case of a Derived Logical Fiberings  $L^{(I)}$ , in which an equivalence relation is defined via the corresponding mediation scheme.

### 2.2.7 Applications

Many interesting applications of the principles of Logical Fiberings are thinkable. As shown in [PFA04], it may be aptly used to model Multiagent Systems. More specifically the case of three cooperating robots is given, where the local state base of every robot is a logical fiber and the collection of the fibers (Logical Fiberings) offers insight on the global state of the system. Moreover, communication between the robots is modeled at the overlap spaces, i.e., where robots cooperate. We do not go into detail, as the application, later developed in this thesis, is not genuinely a Multiagent System. However, it should be said, that in both cases, the principles of Logical Fiberings are explored for data modeling, which might in the future be used in many different intelligent approaches as it naturally provides more expressive data representations through its logical connectives.

## 2.3 Web Access Management (WAM)

Web Access Management defines methods to regulate access to web resources on the Internet. The field may be divided into the management of outgoing and of incoming connections. Also called Internet Access Management, the routines that handle outgoing connections use to restrict the usage of the Internet by employees of an enterprise, students of a faculty or any other user, in order to prevent them from reading material not related to their intended activities or accessing content with immoral or even criminal background. On the other hand it may be interesting to manage incoming connections to an enterprise in order to be able to provide sensitive data to the correct people and prevent all others from these areas; this field mainly employs methods of IAM (Identity and Access Management).

### 2.3.1 Identity and Access Management (IAM)

Identity and Access Management is one of the main concerns of IT-security, according to a survey given by [MAR04]. It deals with the topic of first authenticating the users as they access the network (which may be done by means of passwords, keys or even biometrics) [CHE05]. Secondly, rules are administrated in order to open the correct resources to the intended users and protect the ones, which shall not be accessed.

IAM in the enterprise is not only an issue of pure security, but also of ease of use (as commercial applications use to demonstrate) and return of investment, which [MAR04] gives as one of the main points for IT-managers.

Being a pressing company issue, a great universe of commercial applications exists.

A special situation in this scenario is the remote patron authentication used often by libraries (see example in Figure 2-15 [OHA04]) dealing with issues of how members should access resources from remote places. In this case it is necessary to make the system treat the remotely logged users just like local users delivering the access according to the configured user rights [BCR03].

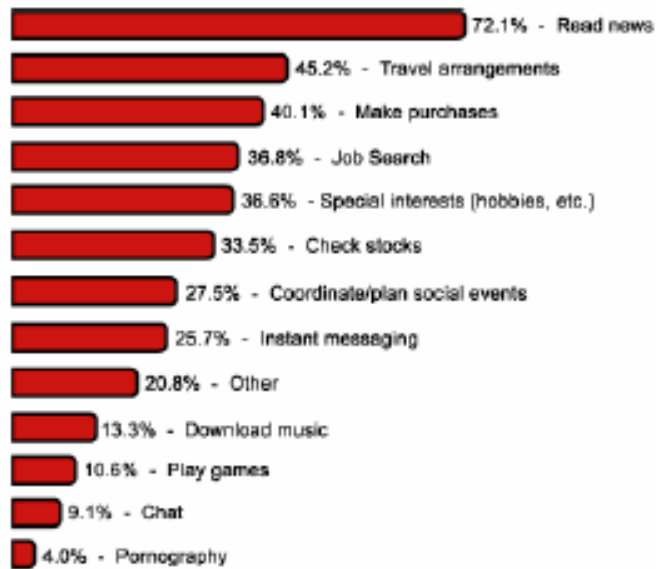


Figure 2-15: Typical remote patron authentication web page of a library

### 2.3.2 Internet Access Management

The management of outgoing connections may be done in order to prevent individuals from accessing inappropriate content, which may have noxious influence on their education (in case of

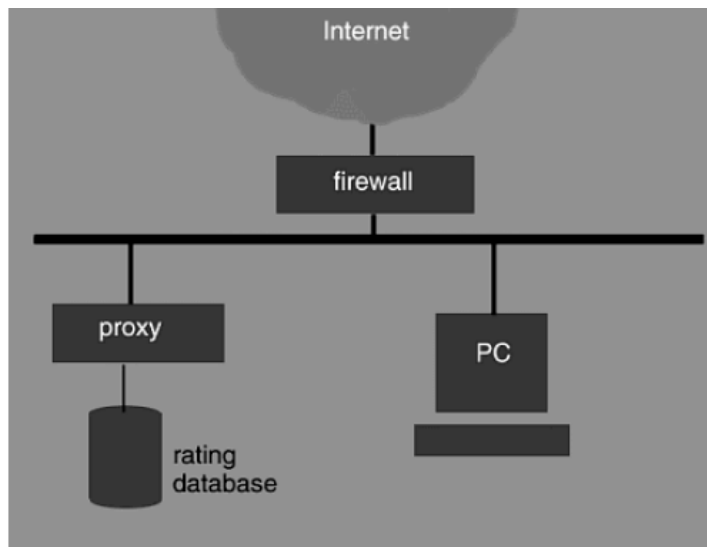
children), their productivity (in case of company users) and/or the bandwidth provided. Misuse of the Internet during working hours is a very wide-spread phenomenon. According to a survey with 451 employees given in [CYB05] (see Figure 2-16) the following percentages applied concerning the access of material not related to work:



**Figure 2-16: Percentage of employees accessing non work-related Internet content**

The survey shows a very high percentage of employees that access content, which actually has nothing to do with their work activities. This scenario may be more or less problematic – depending on the company arrangements. More unnecessary accesses may provide more bandwidth overload or bring viruses and Malware into the enterprise environment.

Therefore most enterprises adopt a scheme that blocks part of the undesired accesses. This is done – in most cases – using the solution similar to the one depicted in Figure 2-17 below [BAK95].



**Figure 2-17: A typical firewall topology**

In this scheme the user does not connect directly to the Internet through his PC, but instead connects to a Proxy-server, which enforces policies for Internet Access, that are saved in a

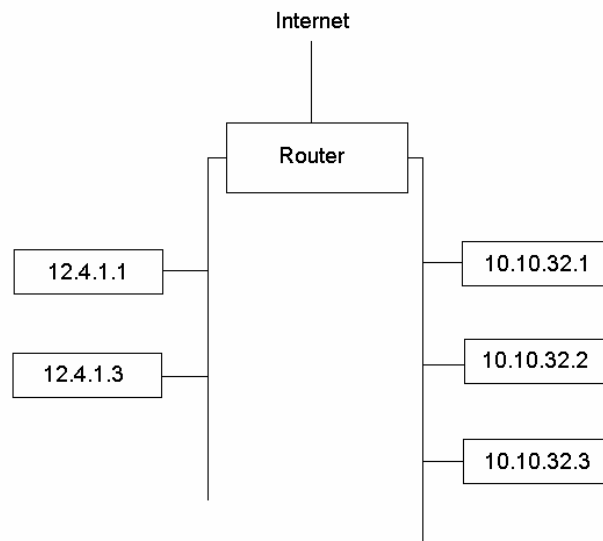
security database. The requests – depending on permissions – are forwarded to a firewall by the proxy. Main task of the rather simple firewall (which in this case might also be a router) is to block direct traffic from clients and only let the proxy requests pass through [BAK95]. Thus there is no way of direct access and security rules actually apply in all cases.

### 2.3.3 Firewall Architectures

A “fire wall” in its classical sense is a wall that is fire proof, used as a barrier to avoid propagation of conflagrations. In its metaphorical meaning which applies to IT-security, a firewall is a device (software, arrangement or equipment) that limits network access [CHE05], generally by filtering traffic.

The following types of firewalls exist [CHE05, WAC02, WIL04]:

- Package filters: Normally implemented at the router of a network, package filters eliminate packages of data from the passing data stream on the basis of its origin, its destination or the port number (see Figure 2-18). No actual context is maintained, instead, all decisions are taken considering the current package only. Depending on the router, this filter may be implemented on the entrance or the exit point of a network, or both. In some cases the ISP provides such technology [FER98]. Basically, package filters are as good as their configuration, and erratically configured package filters may cause more damage than benefit to their network. The security policy has to be very well pondered. Especially configurations with spoofed packages, which represent a major threat to this kind of firewall, have to be taken into account, thus needing very careful engineering for the access rules. Furthermore, traffic of routing information has to be limited in order to avoid inaccessible machines to become accessible through alternative routes [CHE05].



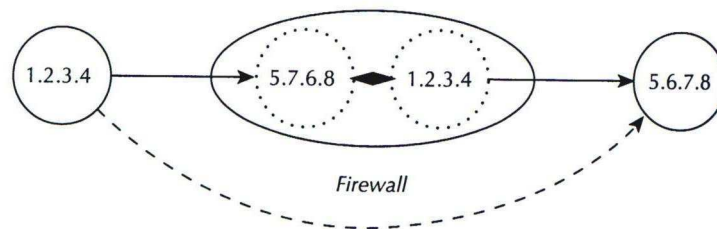
**Figure 2-18: An example of the application of a package filter. Addresses on the left are accessible through the Internet whereas addresses on the right may only be accessed through addresses on the left**

- Application level filters: Application level filters seek to not simply limit traffic on the basis of its way through the networks, but concentrate on the data, which is being transferred. Thus, it may identify several formats, virus code or obscene words and block traffic accordingly. In this way, its main objective is not to be restrictive on addresses, which allows more mobility to users using different workstations. This type of firewall functions by the principle of “store-and-forward”, i.e. the information is first locally stored and then a



decision is taken whether it shall be forwarded or not. Problem of this approach is the intelligence needed at the gateway as well as a reasonable clock overhead, which might be necessary for processing [WAC02, WIL04].

- Circuit level gateways: Circuit level gateways work on the TCP level, retransmitting the information from one server byte-by-byte to the other, i.e. packages do not flow peer-to-peer. This has the advantage of terminating several IP problems, as fragments and firewalking probes at the retransmitting host, whereas the server, for which these packages are meant, only receives well behaved traffic. Still, some controls as delay on specific ports, a list of possible outside callers as well as user authentication may apply [WAC02].
- Dynamic package filters: Dynamic package filters are the most typical and distributed technology for firewalls. Dynamic package filters, just as normal package filters, examine the packages and discard some of them upon their characteristics. This time, however, the semantic of a connection is established, saving the identity of sent packages and letting the answer packages pass through as well. Furthermore some ports (as FTP) receive special treatment. Rules may be changed dynamically (although this might bring obvious problems of users errors about) and the data can be retransferred by the firewall as in circuit level gateways. In order to hide its existence, the firewall may answer with the server address and use the original source address when replicating the package (see Figure 2-19) [CHE05].



**Figure 2-19: Possible package replication with dynamic package filters**

- Distributed firewalls: Distributed firewalls describe a distributed topology, in which every host guarantees the obedience of security rules. The main advantage of this architecture is the elimination of a central controlling instance, which could be a central point of attacks, configuration problems or even hardware failures that might all have a problematic influence on the network traffic, its security and its velocity. Furthermore, when a machine is outside of the normal network circumstances, still the same security rules apply (e.g. notebooks, which are taken to other places). The main downside may be the necessary space to save all this information and the problems of updates on all machines simultaneously [CHE05].

Firewalls may provide a certain level of security, however, if improperly used they may only produce a wrong feeling of the real situation, thus bringing even more serious problems about. Several other complications may be observed for the use of firewalls. Examples are [CHE05, WAC02, WIL04]:

- Firewalls are useless against internal attacks. As they only observe and filter traffic with the outside environment, the internal users must be well behaved. Depending on the organization, this may bring about extensive controls of the staff and their behavior online. Furthermore, visitors may be present in the organization and there might be fewer possibilities to control them. Finally, employees with good notion of the security patterns are a special security problem as they might become victims of social engineering.
- In general, firewalls are useless if the intrusion does not go through them. A disc with a virus or a download of Malware from the Internet may be examples of code, which may be beyond the control of a firewall.

- Compacted or encrypted data could produce serious performance problems. Thus, often a treatment of such kind of data might be moreover an issue of cost-benefit evaluations than of only security-guided principles.
- Trust-relations have a high potential of producing situations, in which trust is not actually freely established between two points, but inherited through the trust with a third party. This may create surprising conditions for all of the involved.
- Not having to do with the concept itself, but rather with the normal quality of hardware and software, firewalls sometimes present bugs and do not function the pretended way. These resulting security problems frequently become notorious issue with hackers and have to be fixed immediately in order to not create an open door for intrusions.

### 2.3.4 Tunnels

Tunnels are mainly encapsulations of messages using several protocols inside each other as well as specific implementations for tunnels (e.g. IP over IP, PPTP, L2TP or SOAP). The objective of tunnels is to move the information through firewalls without the possibility of having part of its contents filtered and – more generally speaking – to protect the information, which is handled, from outside harm, as depicted in Figure 2-20 [GUL00].

Although tunnels may bring about considerable simplifications as there will be no need for defining any exceptions on the firewalls, the problem remains that these may be a serious threat to any security system when being established by two mal-intentioned users. Secondly, tunnels themselves may have several security flaws and the information may be “decrypted” by others in any part of it – mainly at its extremities – thus producing secondary problems [CHE05].

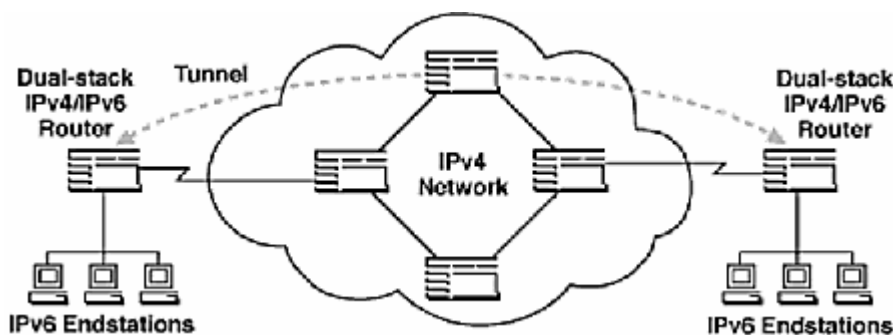


Figure 2-20: Example of a tunnel topology

### 2.3.5 Applications

There is a wide range of commercial products, which are developed and sold mainly to take care of WAM issues in the enterprise. A small choice is presented in this section.

- RSA ClearTrust®: The tool of RSA security has the main objective to protect resource in growing enterprises. It advertises with the simplicity of a single sign-on (SSO) capability of web access management technology which shall enable users to move seamlessly and efficiently between numerous applications and domains once they have successfully authenticated themselves. Access management privileges are tied to user identities, i.e. WAM is closely linked to the issue of IAM. RSA claims an easy and user-friendly management of complex organizations [RSA05].
- eTrust® Web Access Control: This product of Computer Associates International aims at providing an integrated access control solution. It is projected not to disrupt business processes and handle all common business solutions in one step [CA05]. An easy and



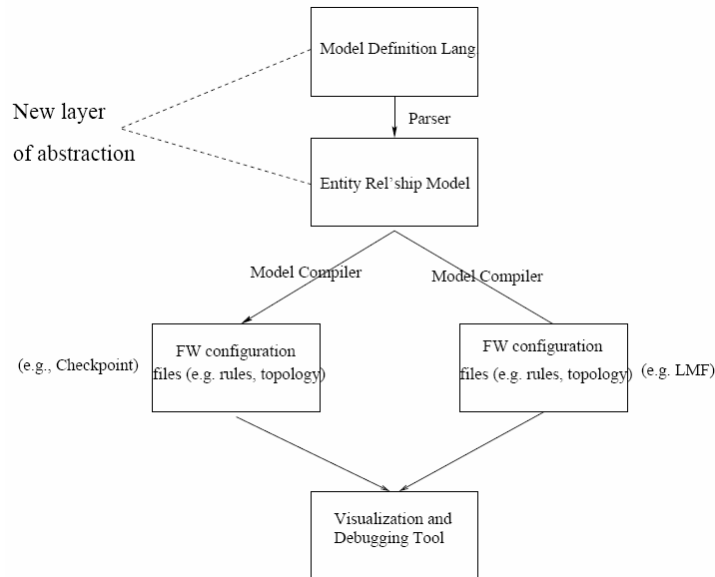
applications. Mainframe application deployment is improved by Java 2, JAAS and Websphere on z/OS. Design is maintained flexible and implementation is possible on several platforms. Identity and security management come in one product, thus reducing costs for installation [IBM05].

Mainly, it may be observed that a large number of products differ a lot in concrete implementation, user interface, connectivity etc. whereas; however, the core logic does not vary a lot. The central tasks remain the authentication of users, the application of roles and the management of resources according to these roles.

### **2.3.6 Scientific Approaches**

Web Access Management and firewalls form a thoroughly commercial domain with a whole range of enterprises competing for the confidence of wary costumers. However, some interesting scientific approaches have also been developed. This section shall give some examples, which address typical firewall problems, yet faced by a lot of companies or even home users with traditional firewall implementations:

- The Roaming Active Filtering Firewall (RAFF), which is presented in [LEH03] is an answer to typical security problems of handheld devices. Firstly, these devices use to have limited capacity concerning bandwidth, CPU, storage etc. Secondly, they are used in a wireless networking environment, which by itself, poses additional challenges on any security implementation. RAFF uses the Lightning Active Node Engine (LANE), where the active node is divided into active server and active router whereas terminals are mobile. The approach introduces the use of several agents: the home agent, the beachhead agent and the correspondent agent, all located at different locations of the wireless network and helps the mobile user by furnishing a technology which follows him, even to remote networks. Thus, the RAFF approach is an adaptation of the existing traditional firewall approach to a new virtual topology, which clearly uncovers one of the main problems faced with firewalls: topology dependency and difficult adaptability to new technologies.
- The firewall management toolkit “Firmato” was introduced by [BAR99] with the intention to treat another problem typically encountered: The great difficulty to manage a corporate firewall rule set in a consistent manner. Firewalls typically demand high efforts of configuration as the rule sets use to be very low-level and have to be re-configured for every instance (i.e.: node on the network). Moreover, any changes in protocols and architectures bring problems of adaptation and a reasonable overhead of service work, which easily leads to security wholes [BAR99]. Because of these problems “Firmato” introduces a high-level concept of roles, which are – after establishment – compiled to the low-level firewall code and replicated to whatever instance necessary on the network (see Figure 2-23). In order to open the rule sets to better human understanding, a rule illustrator was created.

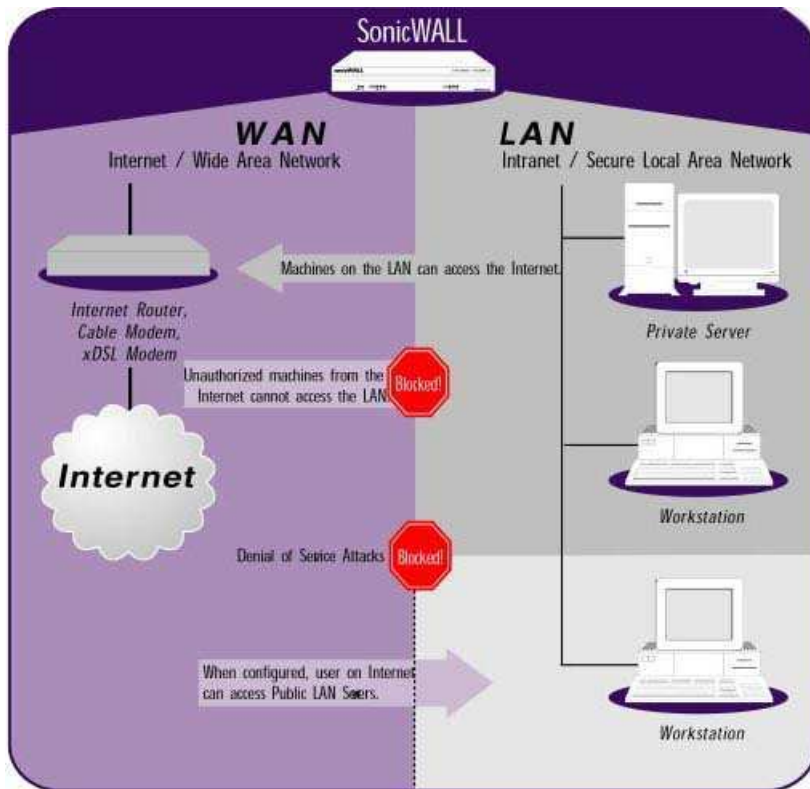


**Figure 2-23: Firmato toolkit architecture**

- Another problem is the number of nodes on an internal network of large corporations. Whereas typically a central firewall proxy treats all outbound and inbound requests, automatically all internal nodes remain trusted. This is a very dangerous trust assumption, as all nodes on a large network cannot be controlled and as it is thoroughly trivial to physically violate this protection establishing new clandestine access points [IOA00]. Secondly, a single firewall instance has the clear tendency to develop into a bottle neck with a rising number of nodes in the internal network. Thirdly, as the rules are held centrally they tend to be coarse-grained and thus not offer the expected protection. In order to deal with all these problems Ioannidis et al. provide a suggestion for an implementation of a distributed firewall [IOA00]. The basic idea is to save and maintain security policies in a central manner, however, leave the enforcement to every individual machine. The developed protocol uses the KeyNote trust management system and the Internet Key Exchange (IKE) for distribution of credentials to the client machines [IOA00].

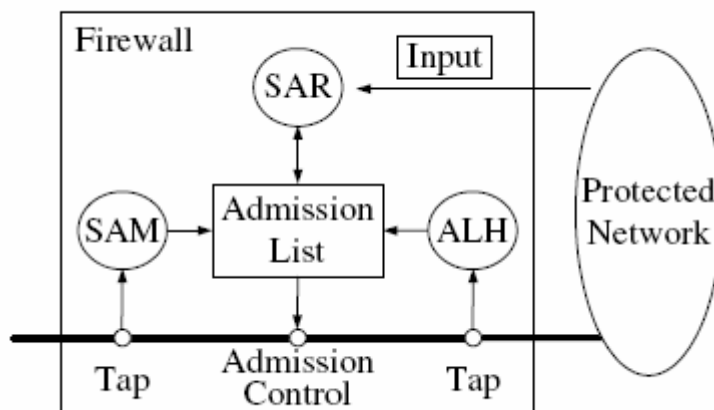
### 2.3.7 Defense against DoS

Web Access Management tools can effectively prevent DoS by managing accesses and, as such, provide access only to legitimate users. Although the focus is normally not DoS, but generally malicious intrusions and data theft, the same architecture may be employed. Basically, to implement a firewall against DoS, one might think of a proxy node which is managing outgoing and incoming queries. All intelligence might be attributed to this node, which, as well, might be implemented by a router or on a personal computer. Some simple approaches of a hardware realization already exist, as, e.g., in the Starwireless G router of Aethra®. The mechanism should be able to keep tracks of all connections going through it and decide whether access should be granted or not. Whereas this technique itself is rather simple, exactly the task of data analysis is computationally costly and difficult to handle. Figure 2-24 would illustrate an example a simple overall architecture of such an implementation, which might be called a firewall (see [SON07]).



**Figure 2-24: SonicWALL® intelligent firewall implementation**

The spectrum of possibly treated DoS and DDoS attacks comprehends all these based on attacks, which disguise themselves via legitimate accesses. These attacks may be characterized as the main focus and real dilemma of DoS and DDoS, because this problem – unlike other intrusion issues - urges for intelligent action and interpretation, posing considerable difficulty on human operators by size and form of data provided (see [MIR04]).



**Figure 2-25: Proposed firewall architecture**

Another example of an approach using a firewall implementation against DoS is given in [XU04]: Relying on the mining of source IP addresses and considering the possibilities of spoofed addresses, the firewall shall block illegitimate accesses. A Source Access Mining module (SAM) stores the addresses in a database, whereas certain threshold values are defined for some

sources in order to sort out congesting clients. The Source Address Refining (SAR) module shall examine the addresses, which might have been stored in an incorrect manner, by sending ICMP echo packets and evaluating the response. Finally the Admission List Housekeeping (ALH) manages validity of entries and collects packet intensity information (see Figure 2-25). Though not effective against all types of attacks, the firewall has shown good performance to protect a network where the bottleneck of processing and bandwidth lies in the network rather than the upstream link [XU04].

## 2.4 Data Mining

### 2.4.1 Introduction

Whereas with every day the size of data available to humanity grows, at the same time the interpretation gets more and more difficult. Large database pose considerable difficulty on anyone wanting to interpret patterns and tendencies “from scratch” working through millions of registers just to find that it is impossible to discover especially the very fine differences between different types of data.

As an auxiliary method to the discovery of knowledge in databases, Data Mining was developed. Whereas initially it was expected that such a method would give automatic or semi-automatic means to the discovery of valuable information [BER97], more recent approaches tend to be more cautious [LAR04, BER00]. Main fact is that the found patterns may be interesting (or not) for anybody analyzing the situation. The mere fact that a Data Mining technique was applied by itself does not yet guarantee a reasonable outcome [REZ03].

Yet, a series of techniques have been developed for this intelligent data analysis, which is also called KDD (Knowledge Discovery in Databases). In many cases these techniques are used in combination with OLAP (On-Line Analytical Processing) or Data Warehouses [LAR04].

The following section shall give an overview over a typical Data Mining process, employed algorithms, chances and inherent problems.

### 2.4.2 Definition

According to [FAY96], Data Mining may be seen as the process of identifying new and valid patterns, which are potentially interesting, in large databases. Thus it may be seen as a means of finding constraints in initially disconnected and raw data. It might, however, be necessary to establish certain formalization even on the raw data as commented in [REZ03]. For such, a Data Warehouse might be a reasonable way, depending on the proposed problem.

Problems, which are typically treated by Data Mining are market basket analysis, risk assessment for insurances up to passenger screenings for flight security (see 2.4.7). The quality of the results is typically very connected to the data quality, which is one of the main issues in Data Mining (see 2.4.3.6).

### 2.4.3 Data Mining Process

There are several approaches (e.g. [ZAI99]), dividing the Data Mining Process into a distinct number of phases.

Following [REZ03], we are adopting a scheme considering the following phases:

- Pre-processing,
- Pattern extraction,
- Pos-processing.

This does not really present a contradiction to other views, but merely is a different, more abstract way to see the same process.

Over the phases of Data Mining several kinds of users interact, who may be separated into three classes [REZ03]:

- Domain specialists, who have deep insight into the treated domain and offer parameters of plausibility needed throughout the process;
- Analysts, who are specialists in the Data Mining techniques themselves and ensure that the process does not get misled by pure technical errors;
- Final users, who shall use and interpret the extracted knowledge. This group of users does not necessarily need to have deep insight into the problem domain itself.



### 2.4.3.1 Problem Identification

As a preliminary phase of the Data Mining process the problem itself has to be identified. During this phase the general parameters shall be defined, i.e. marks to be reached by Data Mining and an initial knowledge of the problem shall be established. The following questions should be thoroughly answered [REZ03]:

- What are the main indicators of a successful process?
- Which performance criteria are important?
- Should the extracted knowledge be comprehensible to human operators or may it be offered as a “black box”?
- Should the extracted data be simple or should it be precise?

In case, several data sources are provided, it may be important to select one, which is most appropriate.

### 2.4.3.2 Collected Data

Basically any data can by principle be processed by Data Mining. Some typical concrete data types are the following [ZAI99]:

- Business transactions;
- Scientific data;
- Medical and personal data;
- Surveillance videos and pictures;
- Satellite sensing;
- Games;
- Digital media;
- CAD and Software engineering data;
- Virtual worlds;
- Text reports and memos;
- The World Wide Web repositories.

### 2.4.3.3 Pre-Processing

In many case the raw data is not yet prepared to be processed straight-forward with a Data Mining technique, because it is not formatted or consumes simply too much space to be loaded. In this case pre-processing has to be carried out on the data, which is done using the following mechanisms [LAR04, REZ03]:

- Extraction and Integration: Data may be available in several formats and attributes. In such cases, a Data Warehouse could be an interesting solution to standardize data sets and offer several different ways of data extraction. Normally, data should be provided best in sets giving attributes and values.
- Transformation: In order to fit the respective Data Mining technique, it might be necessary to define a yet stricter format and transform the data to the inputs required by the specific technique. E.g., data types might be changed, continuous attributes could be mapped to intervals or data might be serialized.
- Cleaning: Noise should be removed by the domain specialist, deleting all completely implausible or invalid data sets. This process is extremely important as the quality of conclusions is directly linked to the data quality and noise might have a great noxious impact on the outcome (see 2.4.3.6).
- Selection and Reduction of Data Sets: Huge numbers of data sets and attributes might turn any Data Mining computationally infeasible. Thus it might be necessary to reduce the number of data sets, attributes or attribute values. The main challenge and difficulty in this case is to preserve the main characteristics of the data. The most frequently used

approach is random choice as it generally provides representative data sets. Another method would be constructive induction, i.e., the creation of a new attribute from two or more others. As a third alternative the number of values could be reduced by changing an attribute from continuous to discrete.

Pre-processing is typically done before data extraction but might also be repeated and the Data Mining process may be executed over and over again until satisfactory results are reached [REZ03].

#### 2.4.3.4 Pattern Extraction

In the pattern extraction phase the Data Mining technique itself is chosen. For this, firstly the type of task to be executed shall be chosen. Mainly two groups of tasks are distinguished: Predictive activities and descriptive activities. Prediction is a way of classifying a new example from the rules obtained by a data set. A typical example of an application might be prediction of credit worthiness used by an insurance company. When treating a descriptive task, the objective is to classify data into pre-defined classes or cluster data into groups, finding relations between the attributes used.

As soon as the task is chosen, the appropriate induction algorithm shall be applied and the data represented in an appropriate way. Typical representations are decision trees, production rules, connectionist models (ANNs), example-based models (k-nearest neighbor, case-based reasoning) or Bayesian Networks [WIT00].

In many cases a single algorithm might not be the best choice. Instead, the same patterns could be verified with several algorithms in order to establish criteria of plausibility [KOH96].

#### 2.4.3.5 Pos-Processing

As Data Mining has its main purpose in the posterior usage of the extracted knowledge, the phase of pattern extraction is not the end of the overall process. Moreover the extracted patterns should be contrasted to the knowledge of domain experts. Especially the difference may be interesting whereas completely different knowledge may raise plausibility issues (see examples in [LAR04]).

Secondly, if the found patterns shall be used for human usage, it may be necessary to develop auxiliary techniques to further reduce the universe of patterns only to the interesting examples. This is true as humans typically are not able to interpret satisfactorily large rule sets.

In the same way, models must be comprehensible to allow interpretation. Thus, simple rules are often preferable to complex and "completely fitting" ones [WIT00].

In case the extracted knowledge does not attend the user, the Data Mining process may be repeated until a reasonable set of patterns is found [REZ03].

#### 2.4.3.6 Known Shortcomings

Several problems may occur during the process of Data Mining, mainly linked to the data quality. Three well known problem fields are (see [ZAI99, REZ03, WIT00]):

##### Data Quality:

- **Overfitting:** The found patterns model the concrete data sets excessively, including all noise and misleading sets. In this case a good abstraction is missing and the results are of low quality.
- **Underfitting:** In this case far too abstract rules are found from enormous data sets. Their lack of expressiveness puts the use of Data Mining in question or urges for more iterations of the whole process (see 2.4.3.3).

- Cleanliness: Most algorithms assume that data is clean and it therefore has to be treated beforehand, which results a high potential of possible errors that might be introduced before analysis.

Performance:

- The data available to human analysis has a very high growth rate. This rate is a considerable risk for any Data Mining approach as algorithms with exponential or medium-order polynomial complexity may encounter very serious performance problems.

Legal Issues:

- Data Mining, when dealing with real life data, frequently touches confidential data and may thus be an illegal process. Privacy issues may be important when considering any personal data research or concluding characteristics about a group of individuals
- The security of the databases may also be a central issue. Theft of personal data or selling of registers may occur and needs to be prevented.

## 2.4.4 Main Tasks

As already mentioned beforehand, two main separations may be made concerning the tasks that are developed with Data Mining: Prediction and description (see 2.4.3.4). Whereas prediction typically focuses on helping with decision processes, description shall firstly produce patterns, which may be analyzed and interpreted by human analysts before being used for later prediction. Data Mining tasks may be further classified into the following [REZ03]:

- Classification: This task shall map input data to a distinct number of predefined classes. Main objective is to find a relationship between the attribute of a class in a way that the class of a new example data set may be predicted.
- Regression: The process is similar to classification; however, a continuous attribute shall be predicted - rather than a discrete one.
- Association: Associations rules define the probability with which the presence of certain attributes implies in the presence of others.
- Clustering: This is a descriptive Data Mining task, which groups data into clusters with similar attributes. These clusters may be disjoint or not, depending on the initial set up and are not predefined.
- Summarization: This task shall provide a form of summary for the data, which may involve basic statistic methods, but also more sophisticated processes to determine condensed rules and data visualizations.
- Dependency modeling: This modeling shall encounter the most significant relations between the variables. Basically two models are used: The structural level, which graphically models local dependencies and the quantitative level, which determines the weight of the dependencies numerically.
- Link analysis: Links analysis extracts correlations between the attributes. Linked data is typically modeled by graphical networks, which contain nodes that symbolize the entities. Link analysis may employ Bayesian networks (see Figure 2-26) [SPI07] or Artificial Neural Networks (ANNs).
- Sequential analysis: From a continuous flow of data, sequential analysis aims at extracting sequence patterns.

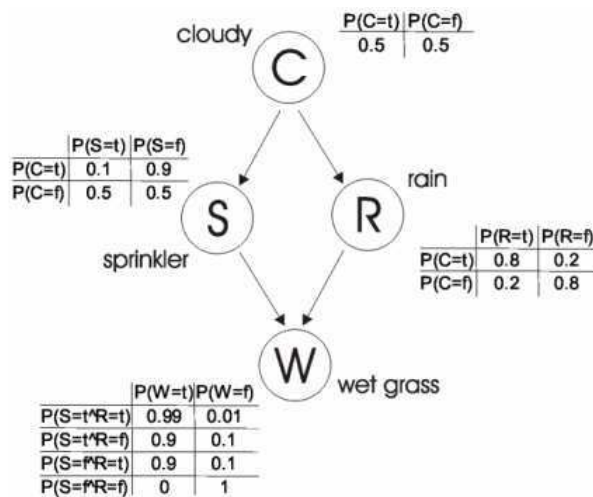


Figure 2-26: Simple example of a Bayesian network

### 2.4.5 Supporting Technology

The Data Mining process uses several distinct techniques. The most important are the following [LAR04, WIT00]:

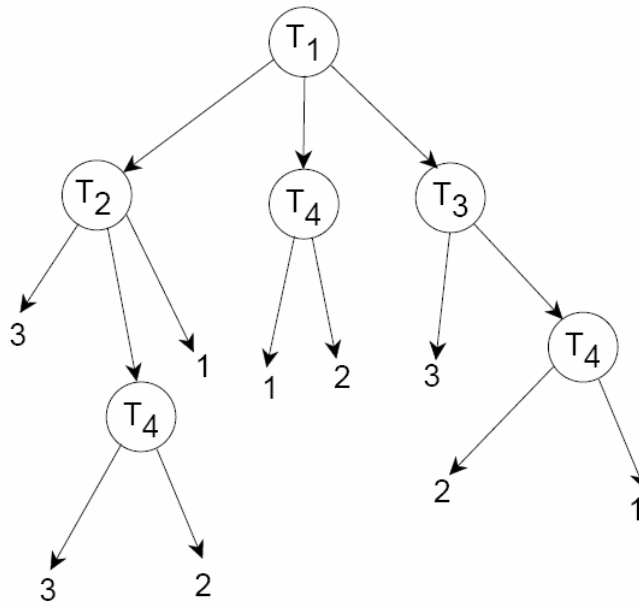
- Machine Learning: This technique is closely connected to Data Mining. It is able to find general rules to concrete data, which may be understood by human analysts.
- Data Warehousing: Data Warehouse are not mandatory in the Data Mining process, however, it highly facilitates the handling of data. Without an appropriate Data Warehouse, reasonable effort may be spent on the pre-processing phase.
- Statistics: This mathematical method examines and characterizes data sets. It elaborates interesting figures, which can help human interpretation. Many Data Mining techniques are based on or originate from statistics. However, statistics normally aim at overall characteristics, whereas Data Mining has its focus on interrelations and comprehension.
- Data Visualization: To facilitate comprehension and reuse of the elaborated rules and conclusions, Data Mining normally employs several forms of graphical data visualization like tree diagrams (see example in Figure 2-27), 3d graphics or spectra.

### 2.4.6 Techniques of Data Mining

Data Mining employs a large universe of algorithms. Basically, several versions of algorithm for the same result may exist.

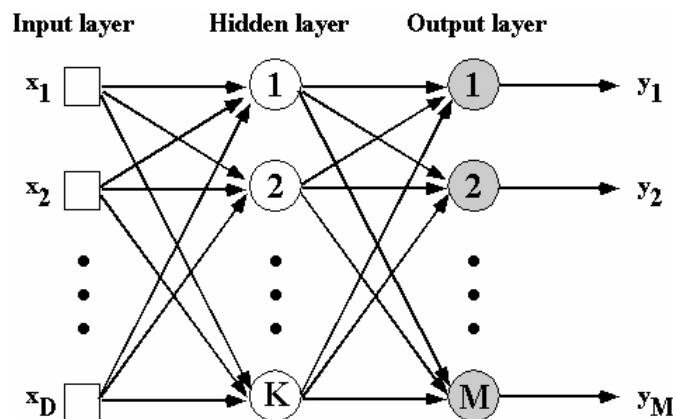
Some typical techniques used with Data Mining are:

- Decision trees: Decision trees are one of the most traditional approaches in Data Mining. Depending on the data treated, they may be called Classification trees (discrete data) or Regression trees (continuous data). The internal nodes of a Decision Tree are test patterns and the leafs are categories (see Figure 2-27). Moving from the root to the leafs, the test patterns are verified and a way to a specific leaf (i.e.: a conclusion) is found [NIL96].



**Figure 2-27: Example of a Decision Tree**

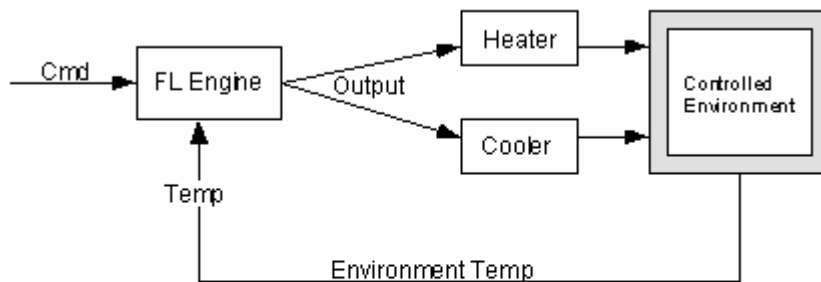
- Artificial Neural Networks: ANNs do not provide humanly accessible explanations for a decision in a certain situation, but yet offer a sound framework to elaborate solutions for prediction and classification in a “black box” manner [HAY01]. A typical straight-forward Multilayer Perceptron topology is given in Figure 2-28 below. It should be noted that a great range of ANN topologies exists and research constantly evolves new models [ROE03, HAY01].



**Figure 2-28: Typical MLP architecture**

- Genetic Algorithms: Using the principles of evolution, genetic algorithms are able to offer sound solutions in unknown environments. Procreation and mutation for successful individuals and death of failures perform an automatic task, which can reach good results with only very few outside guidance [MIT95].
- Fuzzy Logic: Fuzzy logic is an intentionally imprecise framework, which shall model human perception of a situation [KAE98]. Especially control algorithms may be

successfully implemented using fuzzy logic (see Figure 2-29) by defining the whole process based only on vague assertions. Basically, examining a data set, fuzzy logic allows partial membership, which aptly considers noisy, ambiguous or vague environments.



Cmd: Targettemperature  
 Temp: Feedback Sensor in controlled environment  
 Error: Cmd-Temp (+ = too cold, - = too hot)  
 Error-dot: Time derivative or Error (+ = getting hotter, - = getting cooler)  
 Output: HEAT or NO CHANGE or COOL

**Figure 2-29: Simple FL Control System**

- Hybrid Algorithms: There is yet a large scale of hybrid solutions, which successfully mix several of these techniques (see examples: [CAR00, PER02]).

## 2.4.7 Typical Applications

There is a broad range of real life applications of Data Mining. The following list is a choice of some well-known examples:

- Terrorist screening applications: Although Data Mining always played an important role in international secret service activities, it has after September 2001 received a very specific role of helping to find out possible clues to terrorist activities, especially concerning the so-called “sleeper cells”, i.e., units, which are very difficult to detect by normal means of analysis. For this purpose the Terrorism Information Awareness program (TIA) was launched. However, even though it was affirmed that the program only collected legally available data, it was cancelled because privacy issues and some doubt about its effectiveness were raised [SEI04]. The same happened to the Computer Assisted Passenger Prescreening System (CAPPS II), which was to detect possible threats by passengers and typically resulted in further baggage checks of suspicious individuals [SEI04].
- Stock market analysis: In a research undertaken by Alan M. Safer of the California State University [LAR04], data was collected from 343 companies in order to try to predict abnormal stock market returns. The underlying architecture was an Artificial Neural Network. The results provided groups of industries with the most abnormal returns and further analysis led to reasonable results, which were verified by a concurrent application developed with a Multivariate Adaptive Regression Splines (MARS).
- Understanding data from a legal database: Sasha Ivkovic and John Yearwood of the University of Ballarat and Andrew Stranieri of the La Trobe University launched a project, which should find association rules in a legal database with the intention to predict

relations between characteristics of the legal processes. Some interesting conclusions were found, however, they had to be verified with specialists in order not to cause wrong impressions and make potentially discriminating and risky assumptions about distinct population groups [LAR04].

- Prediction of bankruptcies: Due to the recent economic crisis in Asia, several enterprises faced bankruptcy. Frequently, the news came with a certain surprise and the advantage of a Data Mining tool to possibly warn everyone involved of such a possibility, was considered. Tae Kyung Sung from Kyonggi University, Namsik Chang from the University of Seoul and Gunhee Lee of Sogang University developed a project to offer such conclusions [LAR04]. The data structure used should be decision trees. Almost a whole decade with economical data of stable and bankrupt companies was analyzed and a set of rules was provided. The outcome was analyzed and the productivity of capital was found the most important criteria of distinction.

### 2.4.8 Market Analysis

The market analysis given in [MET04] gives a relatively recent analysis of the commercial Data Mining market. Main market focus is understood as the predictive Data Mining processes, which used to concentrate normally on the evaluation of new business opportunities.

It may be said that two groups of enterprises deal with Data Mining issues: Large software enterprises, which see Data Mining as an additional issue for their portfolio (these enterprises frequently might be classified as challenger, as they use to offer less dedication to the issue resulting in lower performance and presence) and specialized software houses, which lead the competition (see Figure 2-30). Expecting further consolidation in spite of the changing technologies, the presence of new comers is every time more unlikely. However, a reasonable variety of tools is thoroughly guaranteed as different approaches heat up the overall competition (compare [MET04]).

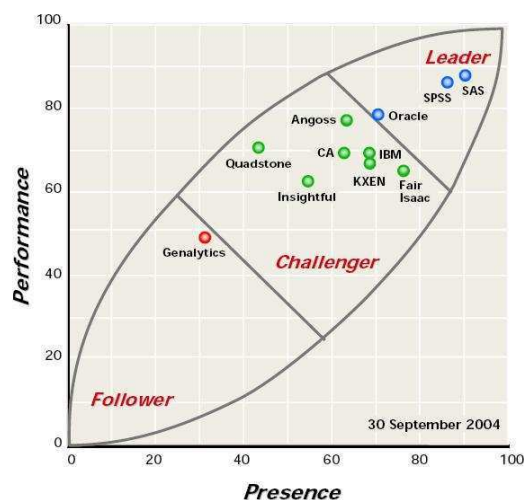


Figure 2-30: Examples of Data Mining Tools [MET04]

### 2.4.9 Future Challenges

Being yet a recent approach of data treatment, the Data Mining process has to face several future challenges to make it become a truly popular and intelligent tool with corresponding market dissemination. Some of these are [LAR04, REZ03]:

- Improvement of the user interface is necessary in many cases. This includes that the whole process must be simplified as the accessibility of knowledge is one of the main goals of Data Mining [PAZ92].
- Database technology on its own constantly evolves in order to offer yet an easier way to access the data sets and create comprehensible reports. This progress generally does not take Data Mining issues into account, which means that Data Mining research faces constant needs of adaptation.
- A true challenge is a satisfactory adaptation to distributed network environments. As standalone solutions get more seldom and even database may be distributed, as a whole, Data Mining has to adjust to it. Especially, the collaborative environment should be taken advantage of and be integrated into the current technologies.



## 3 A Logical Fibering Based Approach

### 3.1 Introduction

Purpose of this dissertation is to present an approach for effective defense against Denial of Service attacks. As seen in chapter 2, during a DoS attack, one specific network node is attacked in order to deny an offered service on the network. These attacks typically occur via numerous connections to a client. Technically speaking, connections can easily be blocked, so there is a basic means of defense. However, the great problem is distinction between attacks and legitimate connections as attackers do their very best to disguise their bad intentions and greatly multiply the connections so that time for analysis gets scarce. Existing defense approaches, which determine if a connection represents an attack or not use to have serious problems of false positive and false negative and thus their benefit may be questioned (they actually contribute to DoS to a reasonable percentage by potentially blocking legitimate connections). This is why an intelligent alternative should be offered. One of the main problems identified with the approaches given in chapter 2, was the fact that they treated analysis superficially, whereas it may appear straight-forward to say that deep analysis is needed to correctly treat the problem.

As a starting point, we consider the data representation as crucial as it forms the very basis of any further analysis (intelligent or not). The superficial level of representation offered by most systems was problematic due to the fact that only high level representations were chosen, which offered serious difficulties of comparison by the different data types (IP addresses, program names, ports etc.) and the mere fact that tiny differences in characteristics, which lead to attack distinction, were not considered. It may be found that all simple ways of analysis are already treated by attackers and their methods and so that even simple DoS schemes are prepared to not leave easily analyzed traces.

Thus, we consider that a detailed and deep data analysis with maximum freedom of comparison was a first step. For this, we propose a fine-grained data structure, which shall save all available data of the incoming connections (which might possibly be attacks). We use a Logical Fibering structure (as demonstrated above) to save the connection data. By the use of two-valued logics, we provide a means of representation, which reduces the obtained information to its primitives and thus offers a deep level of perception. The binary information furthermore offers facilities of comparison as it is natively logic and already "normalized" to a unique data type. Treatment of the data structure inside a computational system, thus, becomes easier.

The raw data inside this fibering structure is yet merely a picture, not yet offering any indication, of which connections present attacks and which do not. This way, a necessity of further intelligent analysis is clear. Out of this reason a Data Mining algorithm was chosen and implemented. It analyzes the raw data and condenses it to rule sets, which themselves shall be used for the concrete decision to block (or not) a certain connection. We make use of a classification rules finding algorithm and an ANN for validation of the found results. To obtain a decision whether a connection presents a threat or not, the current system's health state is analyzed.

The extracted rule sets must be held in the database as "online" storage is computationally costly. Therefore, the same fibering architecture (however, in distinct tables) is provided for the rule sets. They are written to the structure by the Data Mining algorithm. As additional facility, on this structure logical connections may be mounted in order to gain more expressiveness and compactness inside the data. These rules shall be constantly read and contrasted to the current connections in order to determine if a connection is to be blocked.

The present approach is implemented in a system called Fibered Guard (hereafter abbreviated as "FG"), which is a client firewall for the defense against DoS and DDoS.

The following part shall explain and justify the underlying principles and architecture in a more detailed way, contrasting it with adjacent research areas and classifying its way of function. FG is

compared to the existing DoS/DDoS defense approaches mentioned in chapter 2 in order to point out its main benefits. Finally technical implementation details are given to give insight into the system’s method and internal architecture; future development possibilities and plans are discussed.

### 3.2 Motivation

Any DoS and DDoS defense system has to face one main complication, here called the “dilemma of DoS and DDoS defense” and depicted in Figure 3-1. Basically, (D)DoS attacks combine spoofing and data hiding (in several forms) with an overload situation. Whereas spoofing obviously asks for a thorough analysis to serve as a basis for defense against the overload situation, this thorough analysis is prevented by the overload itself as it typically acts on resources, which are needed by the analysis.

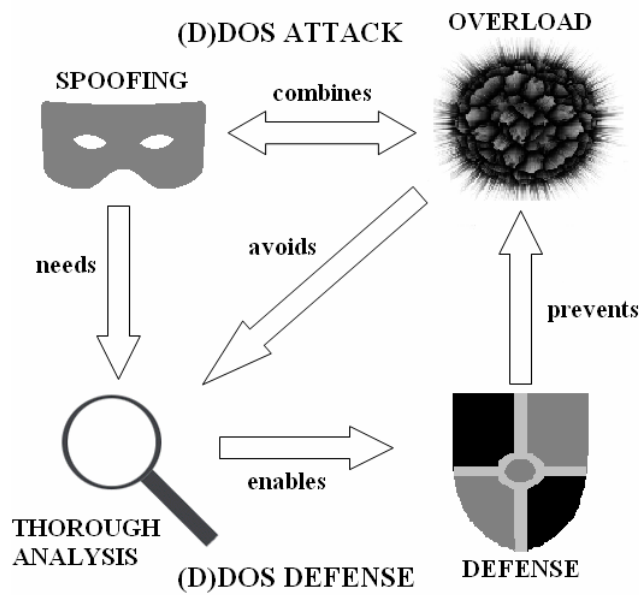


Figure 3-1: The dilemma of DoS and DDoS defense

Motivation of the present approach is the fact that many existing algorithms (see chapter 2: “State of the Art”) for defense are over-simplistic and/or cannot decipher successfully a DoS attack situation. This is in many cases provoked by the (D)DoS defense dilemma as a good analysis is not executed because of performance issues. This has the power to actually extend the DoS situation and help attackers through immature conclusions on an attack situation. As a matter of fact, frequently the main principle of DoS attacks has not be thoroughly understood, which is the denial of a service to a legitimate user (see chapter 2). Our objective must be thus to continue providing the service and not just to “save the machine”.

Out of this reason and to break the (D)DoS defense dilemma, an adequate algorithm must follow four principles (see [SCC05]):

- Obtain a reasonable universe of relevant data – as long as computational resources are available, all possible data must be analyzed;
- Examine data at a very detailed level in order to perceive even the tiniest “different” pattern inside and use this pattern for further analysis. In this sense, analysis of DoS attacks may be comprehended as if it was doing pattern analysis on a picture, namely the image created by the DoS attack scenario, which is surely not a perceivable image to the human eye, but which may be treated by similar technical means;

- A DoS/DDoS defense system must be extensively tested and evaluated before live use, which is due to the fact that this kind of security approach, which blocks connections, has presents high risks through possible false positive and negative, thus potentially creating DoS. It should be noted that testing on live data is very difficult in this domain [MIR04], which is why a solid simulation has to be created first.
- Extensibility, learning and maximum protection of user rights.

### 3.3 Main Algorithm

Firstly, the main algorithm shall be discussed in a high-level manner in order to be able to link the details given in the further part of this chapter. With “main algorithm” we define the way that Fibered Guard defends against DoS and DDoS attacks; it does not comprehend administration, configuration and tuning.

Fibered Guard works the following macro-steps in a chronological order:

1. Polling of connection data: The current connection situation is read and the corresponding data is saved (in the Primary Data Representation). All available data is treated as the system must manage to distinguish an attack situation based on any of the characteristics or part of them.
2. Determination of the system mode: The system works in the modes of “normal operation”, “slight problem operation” and “severe problem operation”. These modes are defined evaluating the use of scarce system resources (as bandwidth, open connections, disc space etc.). For every resource normal, slight and severe problem threshold values are set and the most critical resource will provide the definition of the system mode (i.e. not the overall situation is considered, but the situation of the most critical resource of all).
3. Processing of system mode: First of all, the system mode provides the way the system works, by proving standard ratio of standby operation, analysis or brute force defense (more details on this are given in 3.5.4). Secondly, the system mode is saved to the Fibered Guard together with all of the current connections for further analysis (Data Mining).
4. Data Mining on the Fibered Guard: The Primary Data Representation, which contains the connection data together with the respective system mode, is condensed to rule sets, which are written into the Secondary Data Representation. In the current version exactly one rule is found for problem operation and one rule for normal operation. The Data Mining algorithm analyzes the occurrence of “0”s and “1”s over the set of connections, which took place during normal operation and over the set of connections which took place at problem operation (slight or severe). In case a certain percentage of indication (e.g. the value at “i=12” is equal to “1” within more than 70% of the cases) a fix value (“1” or “0”) is written into the rule. If the percentage is not met, a wildcard is placed. This process is explained in further detail in 3.5.2. Every time rules are created, the connection data in the Primary Data Structure is dropped to free space.
5. Rule usage for new connections: If a new incoming connection meets the specifications of the problem rule and does not meet the specification of the normal operation rule, the respective connection is blocked - in any other case it is allowed.
6. Rule expiration is checked: Rules have a configurable validity. In very fast-paced environments, this validity has to be shorter in order to adjust to the new situations and not stick with old conclusions. When a rule expires, instantly a new rule is created and put into place.
7. Loop: The system’s main algorithm is an infinite loop, which can only be interrupted by the user. This main algorithm is equal for the simulation mode and the real life mode, whereas the simulation mode is executed on a virtual machine and the real life mode on the current PC the system is installed on.

The current algorithm might be criticized as – apparently – a lot of incorrect rules could be created. This assumption could be found considering that there might be attack attempts also at normal operation and a reasonable number of legitimate connections under problem mode (which might thus be potentially recognized as bad and avoided). In fact, the approach works nonetheless out of the following main principles:

1. Need of amplification: DoS and especially DDoS attacks have to greatly amplify the occurrences of attack connections to a victim, in order to reach the wanted impact. Thus, their percentage is high in a well-succeeded attack situation, typically, outnumbering by far the legitimate connections. The Data Mining algorithm adjusts to the majority of cases, thus the “noise” of legitimate connections is ruled out.
2. No success, thus not critical: Attack attempts might occur during normal operation. If they are not succeeding their purpose of depleting system resources, they actually do not form real attacks (the attacker’s mere intention is of no technical importance). Thus the margin of attack connections under normal operation is of no practical importance.
3. Wrong assumptions are temporary: Rule sets have validity, in order to not stick with old rules for changing environments. This way, also incorrect sets may be updated to a more accurate finding over time as they are rechecked in regular, and typically short, intervals.

### **3.4 Module Overview**

In the course of the development of this approach, an application has been modeled in order to prove and validate the principles and offer a practical solution based on the theoretical findings. This system (“Fibered Guard”) consists of four modules:

- The Primary Data Representation
- The Data Mining Conclusion Engine
- The Secondary Data Representation
- The Monitoring Engine

It furthermore makes use of a specific module for validation. The general system overview is depicted in Figure 3-2: FG works as an intelligent client firewall. All connections from and to the client machine, where it is installed, have to go through the firewall. FG saves all incoming data in the primary data representation. This data is examined by the Data Mining engine and condensed to rule sets – saved in the secondary data representation, which is then used to decide whether connections shall be blocked or not. FG constantly monitors the host system’s resources in order to determine when an attack situation is given and when not.

Apart from the fact that the system’s different operational modes (“normal”, “slight problem”, “severe problem”) (see 3.5.4) are adjusted according to the system’s health state, this data is also crucial for the successful establishment of rule sets, as demonstrated later on (see 3.5.2).

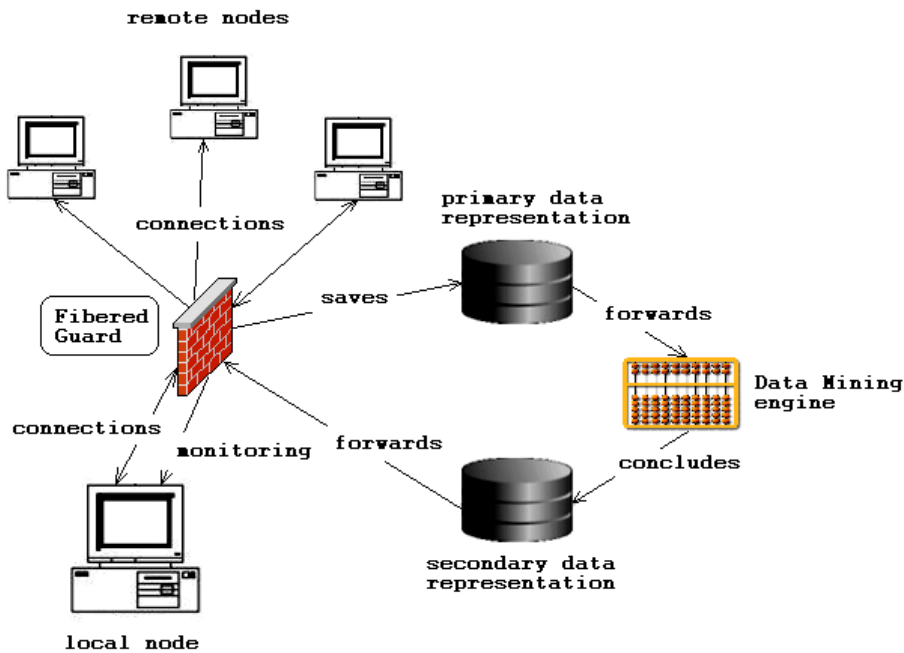


Figure 3-2: FG overview

The following part will describe the modules given in the scheme in Figure 3-2 in their natural sequence of execution, giving their main principles and justification, pointing out limitations and contrasting them to adjacent research areas with similar principles.

FG has a relatively complex and extensive source code, which is why technical implementation details are shown in a more abstract and schematic way to convey the greater picture and feeling of the realization (some more implementation details may be found in the appendix).

Emphasis was put on the application and justification of the principle of Logical Fibering, which is the system's core innovation and fundamentally differentiates it from other approaches.

## 3.5 Description by System Modules

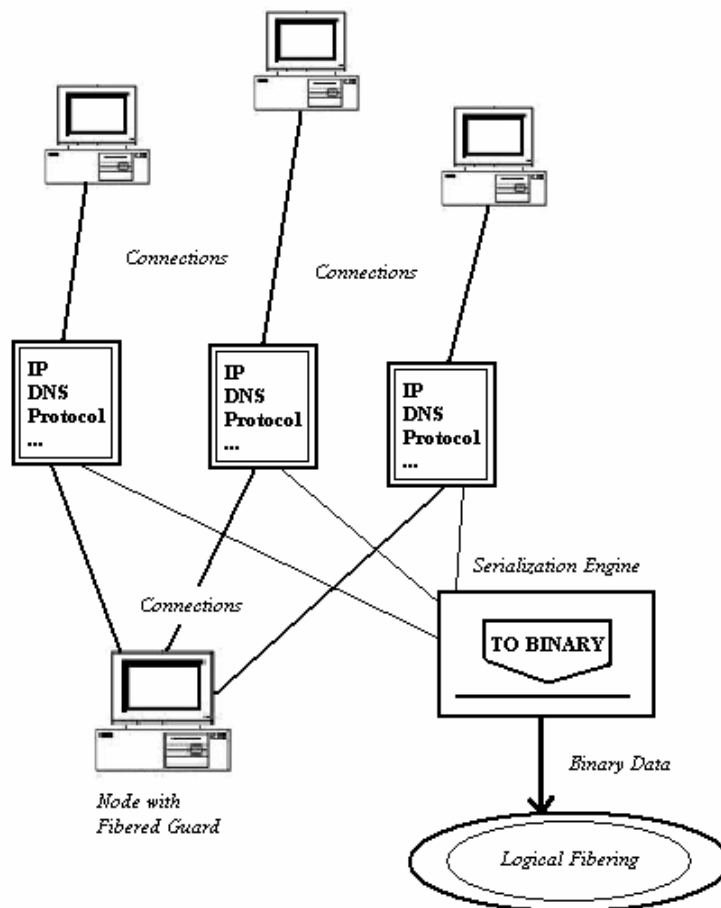
### 3.5.1 Primary Data Representation

The approach takes advantage of Logical Fibering structures for the storage and retrieval of data, which has the advantage of offering an architecture, which reduces the obtained connection data to its primitives and providing a framework for the modeling of logical terms, forming more detailed and computationally powerful rule-sets.

By a formal definition, the present data-representation is a fibered Fibering  $\xi = (E, \pi, B, F)$ . The base space  $B$  contains the indices  $b_{i, i \in I}$  whereas  $I$  is the number of connections in the pool to be analyzed.  $\pi$  stands for the projection map  $\pi : E \rightarrow B$ , being  $E$  the total fiber space and the typical fiber  $F$  a fibering by itself, following  $\xi = (E_i, \pi_i, B_i, F_i)$ . This scheme is depicted in Figure 3-4 below. It may be noted that the fiberings  $\xi_i$  employ two valued logics  $L_{ij}$ . In the yet unprocessed data, these two-valued logics are not interconnected to logical expressions, thus merely offering an image of connections. This is typically denoted as "free parallel system" and resembles the simplest case of Logical Fibering architectures [PFA01].

The process of serialization and storage into the Fibering is depicted in Figure 3-3. The process may be divided in three steps:

- Monitoring of incoming and outgoing connections and polling of connection data. Currently the following characteristics are monitored and used for further analysis:
  - Protocol: TCP/IP or UDP
  - Source IP of with port (if applicable)
  - Source DNS IP
  - Destination IP with port (if applicable)
  - Destination DNS IP
  - Program, which is accessing in case of outgoing connections
  - Connection direction (incoming or outgoing)
- The respective information is serialized. For this process a specific data type (implemented in a separate class) is used, which can read the information in numeric or alphanumeric format, automatically convert the data to basic binary and hold the two values at the same time. IP addresses in this case are converted per octet; alphanumeric characters follow the ASCII table codification.
- As, obviously, the polled information is variable, the entries have to be normalized to the same length in order to be able to use the structure for further intelligent processing. This is done by finding the maximum length and filling up smaller signature with “false”-values accordingly.



**Figure 3-3: Serialization Process**

This raw image may be furnished to a series of intelligent mechanisms for pattern recognition [GAI07], two of which (namely Data Mining and Artificial Neural Networks) are discussed in the following.

The recognition of regularities in a DoS attack situation by Logical Fibering and intelligent algorithms may be compared to finding traces in a picture as both use a low-level representation and strong data analysis engines.

In the current approach, the primary data representation is the first of three phases. It may be seen as a collection of raw data, which is being prepared for further processing, putting it in a greatly detailed representation, which may be further analyzed and from which equally highly detailed rule sets may be extracted.

The fibered Fibering structure has been chosen to aptly save the current connection situation. Whereas on the global level the connections are enumerated, on a local level every bit of gathered information on a single connection is saved. This structure has advantages for processing as it is designed to treat DoS and DDoS attacks. Meanwhile, information is serialized to fit into the binary data structure of the free parallel systems. It, furthermore, is normalized in order to offer always the same length of data string and thus be comparable across the fiberings. This architecture is very similar to those of pattern recognition systems and takes advantage of several principles of this area of research.

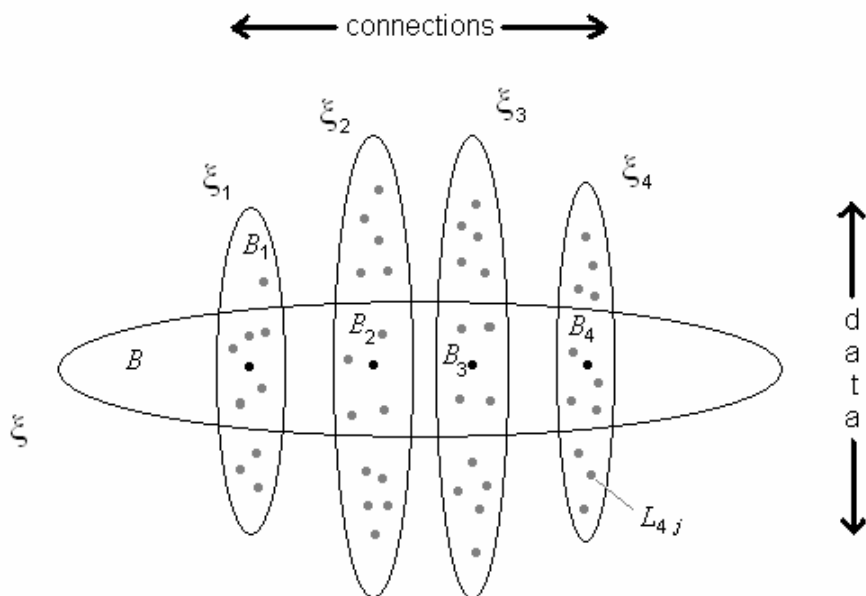


Figure 3-4: Fibering architecture

### 3.5.2 Data Mining Conclusion Engine

Data Mining is a popular way of extracting rules from data sets. It is used mainly to find coinciding circumstances and either make prediction from these circumstances for future situations or classify the examples [REZ02]. In a typical situation a series of conditions are examined and compared with a final result, in order to find the interconnections between the conditions and the results [WIT00]. In the case of FG this processing is necessary, mainly because of two motives:

- Raw fibering data is too difficult to treat directly, the picture of the attack has been obtained, but at this point it has not yet been interpreted. Thus, it is necessary to draw conclusions and use a more condensed set for computational purposes,
- At best, the extracted rule sets shall be open to human analysis. Thus, they should be kept small [REZ02]. A detailed fibering is almost impossible to analyze for a human operator due to its extensive and low-level character (as discussed beforehand).

Inside the present approach the relation between the two valued logics  $L_{ij}$  an attack or non-attack situation shall be given. For this purpose we propose that we – in fact – know how to distinguish an attack situation from a normal operation situation (this is achieved by the monitoring module given in 3.5.4). Parting from this principle, the specific data set obtained during an attack situation is contrasted with those during normal operation and those already extracted to the secondary data structure. On this basis new rules are found and may be directly applied in new situations.

In this approach the following algorithm is proposed, remembering that – at the point when the Data Mining algorithm is used – the respective data is already found in a serialized form in the Primary Data Representation. From there, the Data Mining algorithm shall extract rule sets and store them in the Secondary Data Representation (described below). To achieve this, the following steps are executed:

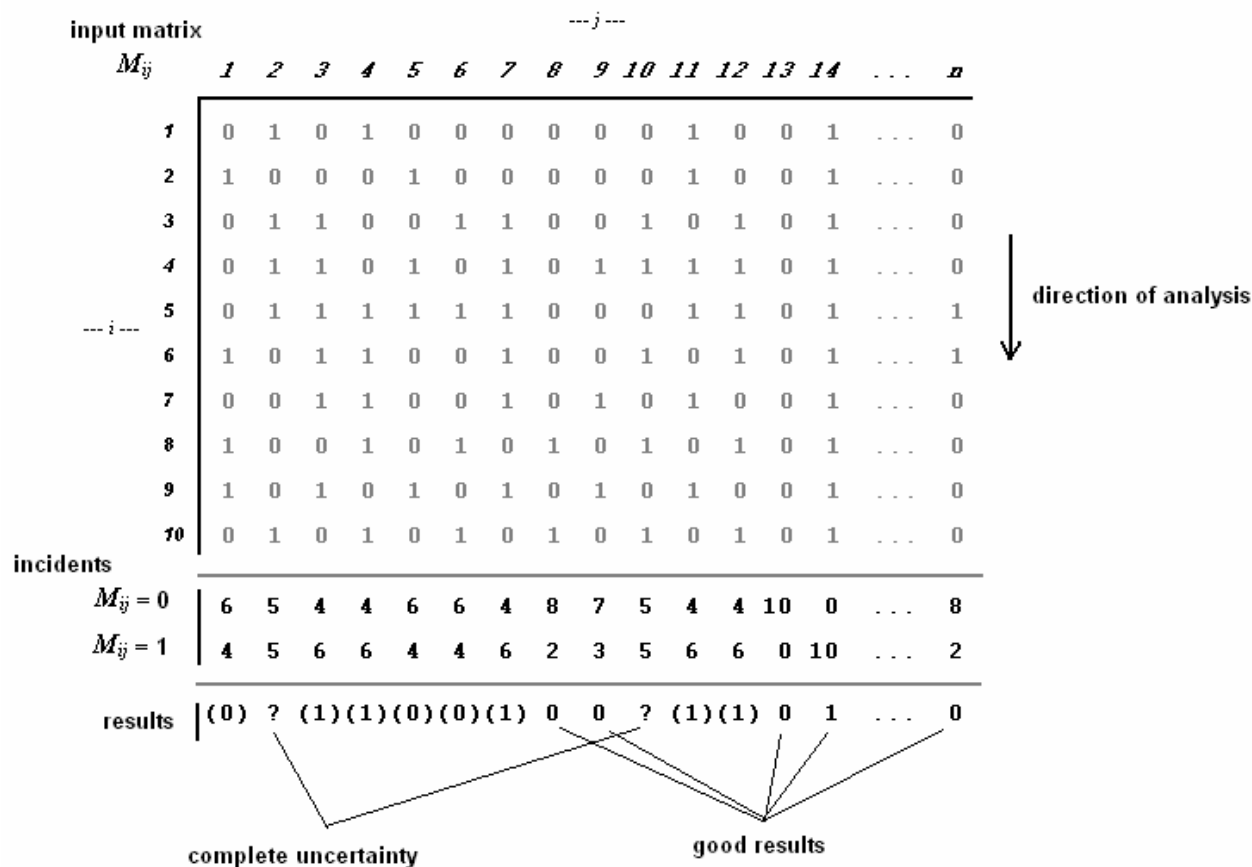
Firstly, the two valued logics  $L_{ij}$  are normalized to a matrix  $M_{ij}$  with  $j$  being of a fixed length  $n$ , in order to enable direct comparison of various lines and values over their position inside the matrix. Each line of the matrix corresponds to a connection, which is modeled by the means of two-valued logics as primitives. Basically, we extract a matrix of variable length for all “good” connections and a matrix of variable length for all “bad” connections. The decision whether a connection is “good” (legitimate) or “bad” (attack) is given by the system’s health state. Thus all connections, which happen during a problem situation, are marked “bad” and all connections during normal operation are marked “good”. This brings obvious problems of marking connections incorrectly (legitimate connections which come in during attacks and attacks during normal operation). However, the following assumptions should be considered:

- Legitimate connections are more frequent during normal operation and attack connections are much more frequent during attack operation;
- In the used Data Mining algorithm the main characteristics are extracted by the adjustments of rules to the grand majority of sets. This is done by allowing  $\frac{1}{4}$  of contradicting connections when forming a rule;
- Attack connections which do not work do not have to be considered attacks;
- Wrong initial assumptions can be repaired by constant cleaning and new assumptions;
- The algorithm shows good results as discussed in chapter 4.

Coming back to the data structure from which rules are extracted, a typical (“normal”- or “problem”-) matrix  $M_{ij}$  might be as shown in Figure 3-5, noting that inside the diagram Truth-values ( $T_{ij}$ ) are depicted with “1” and Falsity-values ( $F_{ij}$ ) with “0”.

The extraction mechanism first sums up all incidents and then makes the part sums for the cases “0” and “1”. Cases in which one value appears 100% of all lines produce the obvious result of such a discrete value in the result set (as seen in column 13 and 14). In all other cases, a measure for minimum certainty (e.g., minimum 75% of the cases as in FG), must be defined. When complete uncertainty happens (50%/50%) or certainty stays below the established minimum, the result is a “wildcard” in the resulting rule.





**Figure 3-5: Example of an input matrix with respective conclusions**

Visualizing the overall principles again with the use of Figure 3-5, the following may be said:

In this matrix several patterns are found. This is done by counting the incidents in vertical direction and deciding for the most frequent incidents. Obviously, there may be good and firm results backed by a strong number of incidents as well as weak results, where the matrix is not very expressive or even completely undecided values (as depicted in Figure 3-5 above).

Generally, it may be stressed that only good results must be used for conclusions. In the above matrix we suppose that all data refers to a single situation (either attack or non-attack) and we are finding a single condensed rule. We may, however, divide the set or amplify it, find more or less rules according to parameterization – this feature is open to user analysis in FG and evaluated in chapter 4.

Currently, FG produces two rules (with short validity, in order to correctly adjust to the current connection situation) inside the conclusion fibering. One of these rules shall describe the typical attack data and its application can be done through an inexact comparison (up to 10 different two-valued logics may occur). The other rule forms the contradiction defining a legitimate connection. The application of this rule follows only exact comparison, not tolerating explicit differences to the newly found connection, which shall be blocked or not – depending on the rule set.

This division of rule sets and the elaboration of two different rules (one for the attack situation and one for the normal connection situation) is depicted in Figure 3-6:

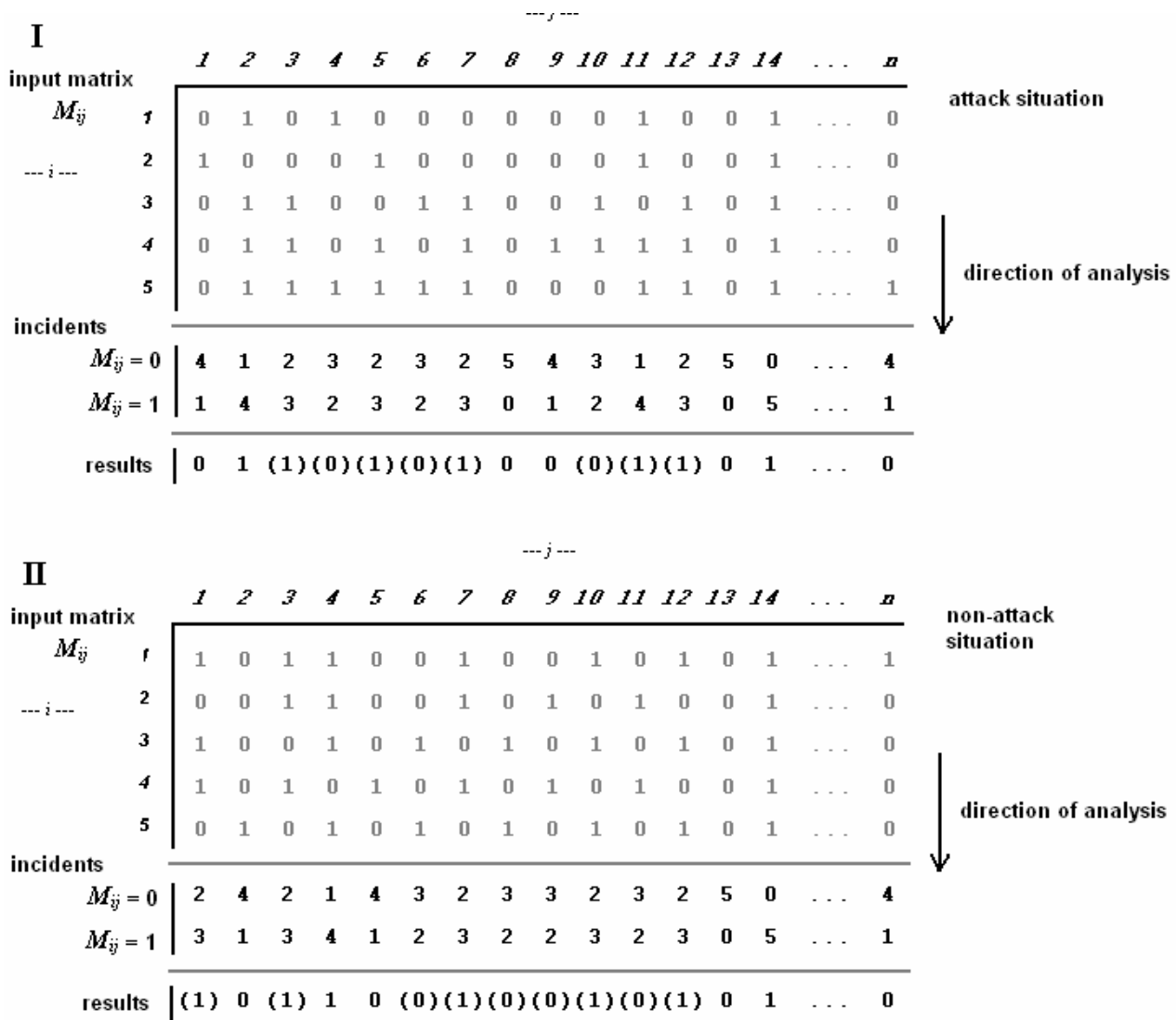


Figure 3-6: Data Mining on distinct sets

Data Mining was chosen as an adequate form to treat the data inside the Logical Fibering data structure, as it may find rules in a computationally inexpensive way by principles, which are open to human analysis (see [WIT00]). Yet, the above chosen principle is one of many possibilities of treatment. In fact, the primary fibering data is not dependent on a specific method of analysis. For instance, [REZ02] suggests that several methods may be used to determine the validity of the results and that good results may be determined by the fact that several methods produce them over the same basis. This is the reason why an ANN approach has also been developed for validation. It is explained in more detail in 3.7.

However, even different views on the same results might be interesting. In our data set, we might want to determine, which types of connection in general occur and open data for human analysis. This could be interesting for the perception of changes in the data universe, which might happen over time. Thus a clustering algorithm could be applied.

A third evaluation could deal with the fact how characteristics are linked between them and find association rules.

Basically, these implementations are possible future developments, but might easily be added out of the modular character of the system, the separate data base and the generality of the fibering structure.

### 3.5.3 Secondary Data Representation

As seen above, condensed rule sets are produced. Each rule set is added to a secondary data representation table in order to be retrieved by the algorithm when necessary. Basically, this retrieval is done periodically and the rule sets are maintained in an internal table (ResultSet) as well as in the database.

Considering a new connection with characteristics  $C_i, i = 1..n$ , we might consider:

$$C_1 = 0 \wedge C_9 = 0 \wedge C_{13} = 0 \wedge C_{35} = 0 \wedge C_{60} = 0$$

This means, that an incoming connection with a *false* value at position 0, a *true* value at position 1, a *false* value at position 13, a *true* value at position 35 and a *false* value at position 60 is indicated for blocking as this data has been extracted from a previous attack situation.

A unique advantage of the fibering structure and its primitives is in this case that a rule set of several rules may be optimized to a more compact and more expressive form, by applying Boolean algebra rules, e.g. the following two rule sets

$$C_1 = 0 \wedge C_9 = 0 \wedge C_{13} = 0 \wedge C_{35} = 1 \wedge C_{60} = 0$$

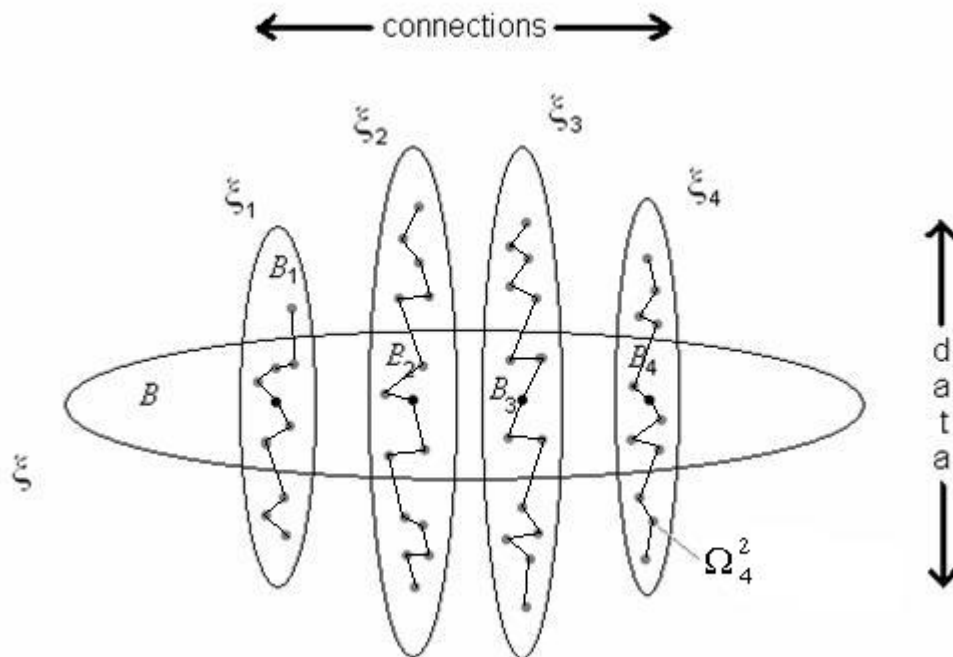
$$C_1 = 0 \wedge C_9 = 0 \wedge C_{13} = 0 \wedge C_{39} = 1 \wedge C_{12} = 0$$

may be optimized to one set in the following way:

$$C_1 = 0 \wedge C_9 = 1 \wedge C_{13} = 0 \wedge ((C_{35} = 1 \wedge C_{60} = 0) \vee (C_{39} = 1 \wedge C_{12} = 0))$$

However, it must be observed that this optimization is computationally costly and produces only readable results when few rule sets are joined. In any case, the idea was to hold this optimization module separate of the operational cycle (it was only to be done in off-peak times). A powerful optimization engine has been foreseen in the architecture, but shall yet be implemented in future versions.

As shown above the Data Mining engine obtains several rules, which themselves can be written to a fibered Fibering, in a condensed picture of the raw data. This secondary data representation is used to furnish the rules for blocking or not blocking connections and shall be accessed in a "shortcut" fashion, offering conclusions to human operators.



**Figure 3-7: Interconnected condensed fibering (secondary data representation)**

It is interesting to note, that by its principles, a Logical Fibering is not limited concerning enumeration (see above) nor by the nature of its logics [PFA00]. This freedom of representation can lead to the expression of complex logical conditions. The Fibering in this case is the framework, the advanced data type, which is used.

By its native functions it may also model communication, which is, however, not used in FG, therefore bringing the need of the implementation of specific functions, which work over the Fibering as the storage and analysis algorithms shown above and the retrieval method, to be discussed in the following part.

The secondary data representation is used together with these processing steps:

- Data retrieval: Firstly data has to be retrieved from the database. This is done by another intelligent online structure, which holds the extracted characteristics in binary format as well as wildcards.
- Data comparison: The data type can be queried through a method, which receives raw data, saves it temporarily for comparison, converts it exactly the same way as in the storage procedure (given above) and compares it to all rules currently held in the data type.
- Blocking: If the blocking rule applies, the connection is immediately blocked. The decision of when a rule applies is configurable via a tolerance rate of not coinciding sets. The main objective of this process is to avoid too close processing, which might produce similar results as those of classical “over-fitting”.

It should be noted that this analysis is only done, if the respective processing time is available (check 3.5.4). How much processing time is used for analysis inside each phase is configurable. A general scheme is explained in the following.

### 3.5.4 System Monitoring Engine

FG constantly monitors the system's health state. This is due to the fact that DoS attacks – by their very definition (see, e.g. [MIR04]) – act on scarce system resources. Not necessarily all of the following resources may be currently used by DoS attacks.

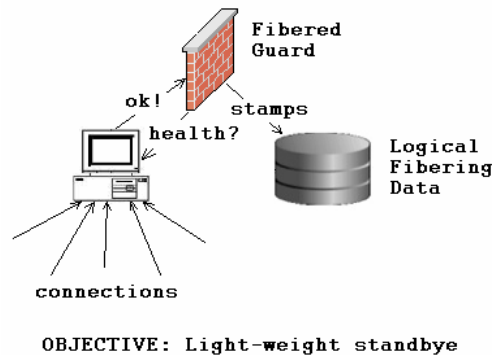
The monitored characteristics are:

- CPU Clock usage: A system, which uses a high amount of its CPU clock cycles, consequently cannot react anymore, neither on the network, nor to simple user requests on the proper client. Therefore, FG monitors CPU usage and sounds an alert if the amount gets to high.
- RAM usage: If the RAM is near to its end, extensive swapping occurs on disc, which considerably slows down any operation. Thus RAM usage is a critical factor, which must definitely be constantly monitored.
- Network bandwidth: DoS attack typically work on this resource, consuming all possible bandwidth, closing all doors for further legitimate connections. FG's central attention is on this subject.
- Disk load: As soon as space on hard disc gets scarce, no more downloads may be made and no data will possibly be saved in swap files. This obliges the client to work at a much slower pace, potentially crashing it "forever" as no disc space is left for temporary files.
- Open Connections: Open Connections are the main target of SYN attacks and other simple and effective ways to attack a network node. Thus they must be monitored.
- Page File Usage: Excessive page file usage slows the client down considerable. Some Trojans work on this vulnerable system point and it might also be exploited by DoS.

Main purpose of the monitoring engine is to decide the correct operation phase and to offer a basis for the extraction of rules with the Data Mining engine.

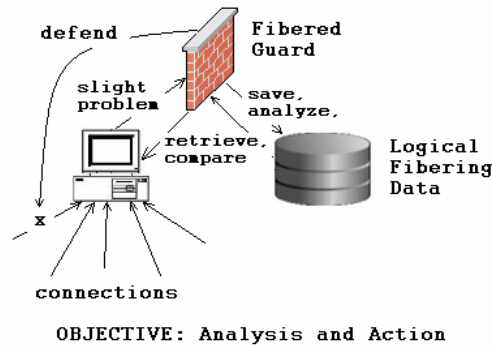
Following the principle of smooth operation (see [SCC05]), FG works with three operation modes:

- Normal operation: Merely constant monitoring is done, to ensure that FG remains lightweight and does not unnecessarily interfere into the system's operation. Only sporadic data collection is done for eventual later analysis (contrast to attack data) (see Figure 3-8):



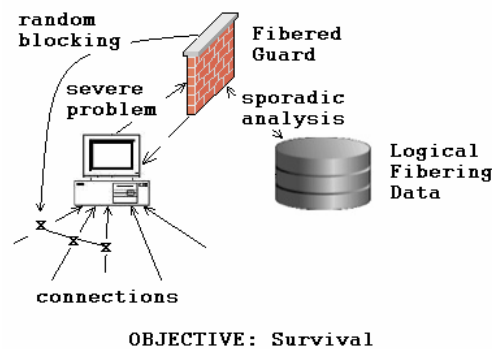
**Figure 3-8: Normal operation processing scheme**

- Slight problem: For each of the characteristics a threshold value is defined (manually by the administration). As soon as the system passes one of these threshold values, slight problem mode is turned on and fibering analysis is done (see Figure 3-9):



**Figure 3-9: Slight problem operation processing scheme**

- Severe problem: In order to stay alive, the system frees resources in a severe problem mode by “brute force”. Analysis is done, according to an initial ratio configured by the system’s administrator (see Figure 3-10):



**Figure 3-10: Severe problem operation processing scheme**

## 3.6 Theoretical Justification

### 3.6.1 General Considerations

We part from the principle that, if any real-life scenario may be expressed by a set of two valued logics at distinct ontological places [PFA01], a DoS attack situation may also be represented this way as it is a subset of a real life scenario. Furthermore, via logical interconnections, which might be referred to as “self-reference” or – in a more specific case - mediation schemes [PFA01], these logics may be bound together to form consistent units. Though these principles appear straight-forward, an important dimension is added, which might be entitled as “molecular” (or primitive) comparing it with the way chemistry views the world’s entities or “low-level”, comparing with low-level programming languages as Assembler. It may appear clear that in a molecular way many of the world’s most complex riddles may be solved and that a machine might be better controlled by Assembler than by a high-level language (taken for granted that the developer knows the operations thoroughly). Thus, it may be stated that Logical Fibered can aptly model connection data, as basically it may model any real-world scenario over sets of two-valued logics which comprehends a logical fibered representation of reality (on this serialization process, see Figure 3-3).

However, there is the question, why Logical Fibered data structures suite exactly DoS and DDoS attack situations. The key lies mainly in the architecture of the attacks: On the one hand the attack might be launched from a local machine, on the other there may be several agents

instructed to attack the same time (in DDoS attacks). This structure urges for local and global reasoning done at the same time: local for single DoS attacks, where the attack might possibly be traced back to a single machine and global for DDoS attacks (see Figure 3-11) [SEC03], where the sole connection is of no use, but a collection of several connections must be analyzed. The typical attack architecture is very well handled by a fibered Fibering, as shown and explained below.

Secondly, any situation on a DoS attack may be mimicked, data may be spoofed, and nothing can be fully relied on. The only clue may thus be a very tiny bit of information, hidden in an overhead of noise and spoofing. Traditional algorithms not only do not manage to fully capture the “picture of DoS”, but also tend to do too much high-level analysis. Logical Fibering has the power to save data in a more detailed way (as shown above). Thus it has much more chances to actually model the necessary information for afterwards equally low-level analysis. These are the main justifications of Logical Fibering as a data structure for the representation of an attack situation. However, it is also adequate for further intelligent processing (Data Mining) as shown later on in 3.5.2.



Figure 3-11: Typical DDoS attack architecture

### 3.6.2 Pattern Recognition in Logical Fibers

The picture established and formed by Logical Fibering is passed on to an algorithm, which may be understood as a pattern recognition algorithm:

- In both cases a conclusion shall be drawn from raw data, a pattern shall be found. In case of an image it may be a face, a fingerprint, a car number plate etc. In case of the overall attack data, intrusion attempt patterns shall be detected.
- In both cases processing is not trivial and has great changes of producing incorrect results (false positive and negative) due to the following:
  - Different characteristics of the same object:
    - Image recognition in images: Different angle, different position, different colors, haircuts etc.
    - Attack recognition: Different IP addresses, different routers, different rates, different consequences etc.
  - General noise, mistakes and incompleteness:
    - Image recognition: Photograph inadequacies, dust, other elements
    - Attack recognition: Good connections, insignificant data
- Performance issues are central for the algorithms as in both cases huge data streams need to be processed and results need to be provided in time. For pictures this is

especially true when considering pictures in movement (movies), which can have up to several gigabytes. A DDoS situation by its very characteristic of depleting the system resources also tends to produce a great universe of connections at once.

- Data representations of images use to be low-level and compressed. Basically, the picture is formed by a great collection of pixels (i.e. picture elements), which in a disjoint fashion do not really seem to make sense. Only with a global vision (“at a distance”), they form the element, which should be depicted. In the same way, raw DoS attack data looks very confusing to any analyst. An overview has to be gained on this data and whereas this might seem absurd from a human point of view, a machine can actually “view” a DoS attack situation the same way as a picture and the same methods may be applied as long as the data is reasonably low level. In fact, it does not matter if a “0”, e.g., means black in a picture or means part of a source IP address. This insight is one of the basic background principles of this project.

In order to better visualize the general function and classification of FG, we might yet mention two analogies:

FG’s approach is similar to that used in syntactic pattern recognition by reducing the obtained information to primitives and working the data with an intelligent algorithm. Another problem treated in a similar way may be that of evaluation of seismic information, mentioned in [HUA02].

Figure 3-12 depicts the typical architecture [SAN06, TOH06] of a pattern recognition system, with the following phases. Each phase is contrasted to FG in order to comprehend its function as pattern recognition system and give a more systematic insight into its architecture:

- Sensing: The raw data is read out by the sensors and compressed to a sensor dataset. FG collects this information of incoming connections by its monitoring module. A set of selected information is kept for analysis.
- Pre-processing: This step comprehends the filtering procedure for new data. It may furthermore be segmented, i.e., pre-directed to a certain set of classes [TOH06]. The result of the pre-processing is called the “pattern vector”. In FG the pre-processing is the serialization of all information and its storage in the Logical Fiber structure. The obtained “pattern vector” is the fiber structure, which can also be understood as a vector structure.
- Feature extraction: The information in this “pattern vector” structure can be rather huge. For instance, in a pattern recognition problem involving a picture on a normal computer screen the information may have over a million components [TOH06]. In FG the same problem happens as the primary data structure is prepared to hold a huge series of registers, but was never by conception meant to use the data directly in a raw form. The Data Mining conclusion engine performs the task of feature extraction: It condenses the data to its necessary minimum, it “selects features” for later processing. Product of this feature selection is the “feature vector”, a compact representation of the raw data. In FG this is the secondary data representation with the extracted logical rule sets. In face recognition this representation is called “eigenface”.
- Classification: From the model, which was created, one might proceed to the classification itself. Taking the base of the feature vector, the most appropriate class is found and provided. The class, in the case of FG, is the decision to block or not the specific connection.
- Post-Processing: Based on the classification, typically a concrete action is taken. Furthermore, context constraints should be applied in this phase. FG firstly blocks the suspicious connection; secondly it is open to human interaction for manual constraint enforcement.

Pattern recognition systems make use of supervised, unsupervised and reinforcement learning.



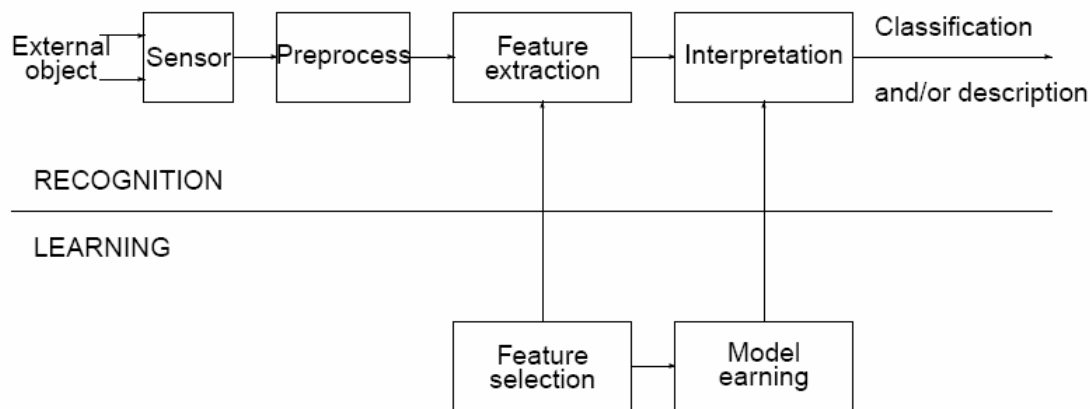


Figure 3-12: Typical Architecture of a Pattern Recognition System

### 3.6.3 Treatment of Typical DoS Attack Challenges

Some specific challenges for DDoS attack prevention have been given in [KUM06]. The Logical Fiber architecture treats these problems the following way:

1. Large number of unwitting participants: If an attack is successfully identified, the picture of DoS or DDoS is deciphered; a message to the participants of the attack would not be a high technical problem. For the defense itself, however, this point has no relevance.
2. No common characteristics of DDoS streams: This challenge has to be corrected. Actually, what should be stated is that there are no easily identifiable common characteristics, especially for a human operator. To automate the identification is the central argument of the present approach.
3. Use of legitimate traffic models by attackers: See point 2. The spoofing and hiding of information is possible, but this approach considers that it is not possible to hide every information, which could lead to a reasonable analysis.
4. No administrative domain cooperation: This should surely be fostered, but is of no relevance for an acute attack.
5. Hidden identity of participants: This point is the same as 3. Actually, any approach should think about defending first and ask questions about identities afterwards. This type of information is very tricky to get in any attack situation and might require social engineering rather than any technical “digging”.
6. Persistent security holes on the Internet: As shown beforehand, the very architecture of the Internet is not safe; it was initially not projected for the immense global network it has become nowadays. Although there are tendencies concerning advances (Internet-2, for instance), these are not technically easy to launch as the number of nodes is enormous. This insecurity, thus, will remain, and leads to the fact that a tool must adapt itself. The present approach gives an algorithm, which does not part from conclusions, but concludes by itself. Thus, new situations are expected and – by principle – already treated.
7. Lack of attack information: A lot of information is indeed missing, but, yet again, not all information, which might lead to the identification of an attack. And in case there was no information at all, actually no tool – not even an intelligent one – will arrive at conclusions. In other words, any intelligent algorithm should pre-suppose that there is “a solution to the riddle” and not search for anything, which is a priori impossible to conclude or make wild guesses.
8. Absence of standardized evaluation and testing approaches: Quite truly, this is one of the main problems with development. There is no good way to test a DoS / DDoS defense approach. The only way to arrive at valuable conclusions is to develop a proper

ambience, a simulation, which is as close as possible to reality, obtain results and afterwards deploy the project into a productive environment, to do the final tests “on the job”.

## 3.7 Comparison and Validation

### 3.7.1 Use of Artificial Neural Networks

Data Mining and Artificial Neural Networks are known for being able to find patterns in apparently odd data. Thus, both methods were studied for use with the primary data representation given above. Artificial Neural Networks seem to be perfect for processing the Fiberling data as translation to the inputs of an ANN is straight-forward. The only lack of an ANN usage is the facts that an ANN presents itself as a “black box”, i.e., not open to human analysis. This is why the present approach employs this intelligent architecture as a method of validation only. Both methods were contrasted and results are given in chapter 4 (Results).

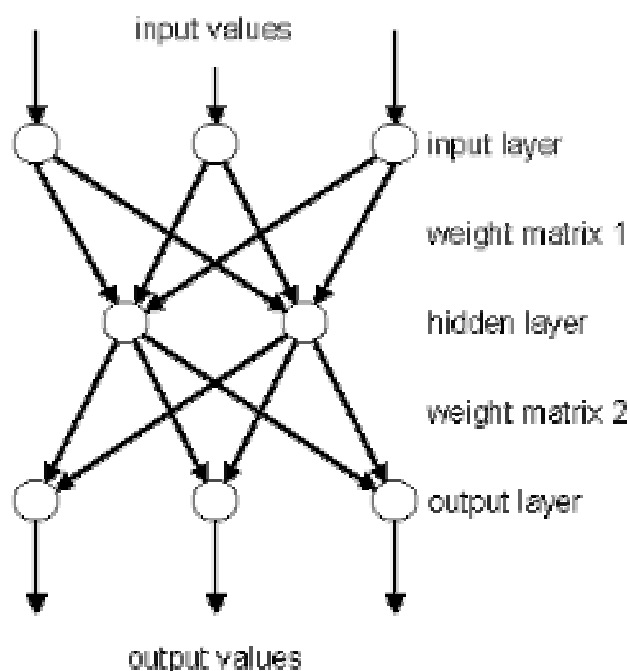


Figure 3-13: Typical Multilayer Perceptron

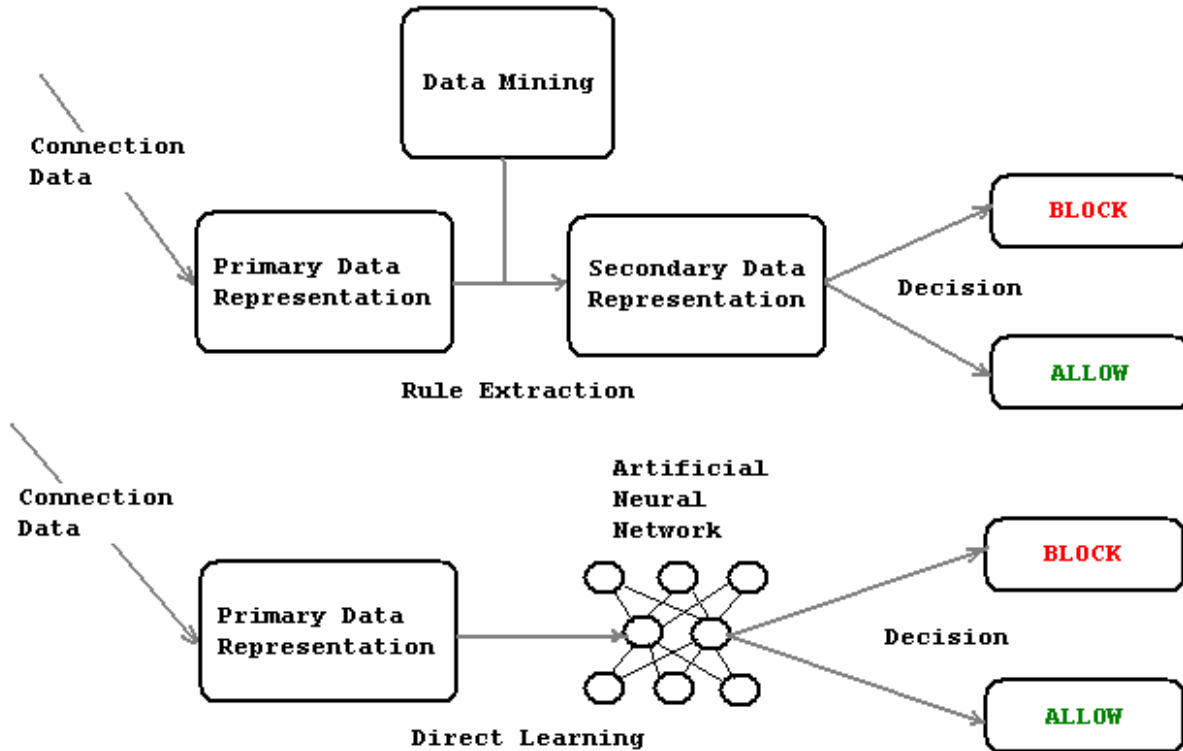
Figure 3-13 above shows a typical structure using a Multilayer Perceptron (MLP) [FRÖ97] which is notorious for its usage in pattern recognition systems [HAY01]. However, it should be stated that the ANN does not produce rule sets and thus the power of the secondary data representation cannot be developed from it.

Fibered Guard uses such a basic MLP architecture with three layers and the following parameters:

- 1480 inputs, corresponding to the maximum 1480 binary values which may be produced by the data of one connection over the Primary Data Structure;
- 22200 ( $1480 * 15$ ) synapses between the input layer and the hidden layer;
- 15 hidden layer neurons, all results were obtained with this value (configurable);
- 60 ( $15 * 4$ ) synapses between the hidden layer and the output layer;
- 4 real type outputs corresponding to the four state possibilities in the fiberling (normal, slight problem, severe problem, upon dos);
- learning rate at 0.35 (configurable);

- activation: sigmoid function;
- learning algorithm: standard Backpropagation;

Meanwhile, it was considered interesting to have a second benchmark for the evaluation of the attack situation. Whereas similar results would indicate that both algorithms were valid, a particularly large difference meant an urgent need to continue investigation – at least a plausible explanation for this had to be found. Further details are given in chapter 4.



**Figure 3-14: Standard Data Mining Processing and ANN validation**

Figure 3-14 shows a comparison of the two operation modes: On the upper half FG's standard architecture is given, using a Primary Data Representation (Logical Fiber) to save the data of incoming connections, a Data Mining plugin to extract rules, which are then saved into the Secondary Data Representation (Logical Fiber) and used to decide whether a new connection shall be blocked or not.

In comparison it should be said that the ANN learns the data sets with their outcome directly from the Primary Data Representation and uses the acquired knowledge to decide on new connections. It should be commented that in dynamic environments a periodical re-initialization of the network might be handy to avoid the creation of strong conclusion tendencies. This, however, has not been implemented in Fibered Guard as the ANN was merely used for validation over shorter periods of time.

At this point it should be mentioned that symbolic and sub-symbolic approaches have different strong points. As pointed out in [WNE94], symbolic algorithms with constructive induction might best learn disjunctive normal form (DNF) concepts, whereas sub-symbolic approaches and symbolic approaches without constructive induction were best on non-DNF type concept. The differences found in [WNE94] were sometimes quite clear as the error rate evaluation on rule representation concepts shows (Table 3-1).

According to [WIT06] one of the main advantages of connectionist algorithms are their robust learning algorithms, which automatically adjust the network to the desired output, whereas such processing is not feasible manually. Through the slower and more continuous adjustment to the

training set it may be considered that it is simpler to avoid over- and underfitting, as ANNs deliberately need much iteration to produce the desired result. This, however also implies in a high computational effort for training, whereas the usage afterwards is very fast and straight-forward.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Genetic Alg. (CFS)	49	45	51	48	41
Neural Nets (BpNet)	35	26	12	22	12
Decision Trees (C4.5)	3.1	2.8	2.5	2.5	2.5
Decision Rules (AQ15)	2.6	2.2	2.0	1.6	1.6
Decision Rules (AQ17-HCI)	2.4	2.0	1.6	1.6	1.6

**Table 3-1: Comparison of symbolic and sub-symbolic methods on a DNF-concept task**

### 3.7.2 Comparison with Approaches in Literature

It should be interesting to further define scope and advantages of the present approach by contrasting it to the main streams of defense systems. For a detailed list of examples of such systems, please refer to “Chapter 2: State of the Art”.

#### 3.7.2.1 Catalogue of Recommendations

Whereas a simple catalogue of recommendations may mitigate the effects of DoS and DDoS attacks and is common in many enterprises, these approaches require mostly a high amount of manual intervention. Thus the effectiveness is completely dependent on the operator’s knowledge in standard cases and absolutely untreatable in advanced situations with high data volume and confusing information [MIR04].

FG implements a semi-automatic solution with the following core characteristics:

- Recognition of difficult structures, which is achieved through the Logical Fibering structure that receives the incoming connection data and the intelligent analysis algorithm;
- Few human intervention as the functioning is projected to be automatic and lets the administrator concentrate only on a few tricky cases whereas the gross part is treated by the system (more on this in Chapter 4: “Results”);
- Possibilities of human interpretation through rule establishment, this means that the system’s actions may be interpreted, which is achieved by the mining of rule sets (explained above).

#### 3.7.2.2 Basic Approaches

Though of striking simplicity, many basic approaches have very clear flaws, because their logic is too straight-forward. These approaches may be seen as “first aid” under attacks, but cannot prevent an attack situation on a continuous basis as they are easily “worked around”.

FG offers a more complete framework whereas both DoS and DDoS attacks are treated, leaving as less leaks as possible. This is accomplished by its generic approach, which analyzes attacks by their effects rather than by a pre-fixed set of characteristics.

### 3.7.2.3 Tracking Algorithms

The main sense of tracking algorithms is to find the attacks source. They are not classic defense systems. Moreover, defense mechanisms have to be yet implemented, as separate modules. Or, eventually, the user might want to take legal action against the source, which is not always possible or successful (as seen above).

Fibered Guard is not preoccupied with any tracing as it is too costly under DoS conditions (using scarce resources for tracing while not defending is not an adequate answer to attacks!) and will not prevent the attack's effect. Especially, data collection will not be possible any more after successful DoS. Thus, it is imperative to treat the attack first.

### 3.7.2.4 Network Congestion Control

Network Congestion Control is a blind and "ignorant" method of dealing with DoS by delegating the problem to the users. Though it keeps the network running, it does not keep the resources accessible, which by its outcome is the same from the client's side, as to bring all the network down. Thus, it is necessary to find a way to distinguish between good and bad accesses and treat them accordingly. The main purpose of Fibered Guard is pattern detection in a noisy attack environment.

### 3.7.2.5 Intelligent and Advanced Approaches

The present approach is an intelligent approach and shall therefore be contrasted to all of the given intelligent defense systems in details:

- A Data Mining Approach to Intrusion Detection: Firstly, the outcome depends – as with all Data Mining systems – highly on the quality of the obtained data. Fact is, that no reasonable pre-processing is shown and that the data structure might not be fitting enough for the DoS situation. Apart from the Data Mining conclusion engine, Fibered Guard makes use of Logical Fibered structure for the treatment of DoS and DDoS, which shows adequate for the registration of connections.
- Mobile Agent Applications: Basically, Fibered Guard might be extended to a distributed agent architecture in the future, thus taking advantage of the distributed behavior of agents. However, installing the firewall in a series of network nodes, almost the same architecture may be reached. Flexibility is given through Fibered Guard's internal characteristics.
- Automata: Basically, automata are a representation of the attack situation, a visualization and formalism to comprehend to raw attack data. Being relatively rigid in structure, this conception has the obvious flaw to put great demands of memory and CPU on the client system, in order to – probably recursively – form the automata structure. FG is by its architecture a less computationally taxing system. Performance results for comparison may be found in chapter 4.
- Neural Networks for Intrusion Detection: FG offers an ANN algorithm for testing and validation. Main criticism of this form of technology may be the adequate choice of an ANN and especially the fact that ANNs do not use to work well if their inputs and outputs are not aptly projected, by balancing and normalizing them [SCR03].
- DDoS defense by offense: Conceptually interesting, this approach does not consider the fact that actually "speaking up" potentially creates DoS. It must be matured. One of FG's main principles is, on the contrary, smooth operation, i.e. every change in configuration, any action taken must be result of thorough evaluation. This is never trivial and must before most not create DoS.

## 3.8 Implementation

The following part shall give implementation details in order to convey a picture of the system's functioning and interior architecture. As classical system documentation would surely go too far for this thesis because of the lack of expressiveness, this part concentrates only on the main principles of the system and its core architecture characteristics. Some more details and a short user's manual may be found in the appendix.

### 3.8.1 Main Directives

Main system directive is smooth processing, which means that the non-trivial task of achieving balanced answers to DoS attacks was put in evidence. By the fact of simply mitigating the effects of the attack and dropping connections randomly or by a specific ratio, many legitimate connections are dropped and in the case of an e-commerce site these frustrated users might think of using a more confident site and never come back. Thus connections must be treated as precious and held as long as possible and as long as there are no good indications of a DoS attack. This is achieved by the intelligent algorithm and very detailed analysis shown above and the implementation of the three system modes: normal operation, slight attack and severe attack. Secondly, an Anti-DoS system only works well on a monolithic approach as every gap brings the necessity of other systems, which possibly interfere and pose yet more performance demands on the host machine (which is also potentially creating DoS and thus must be avoided at all costs). FG is an approach, which defends against DoS and DDoS alike by analyzing their outcome (effect on the system) and scanning the "connection picture".

### 3.8.2 Feasibility Issues

Elements in Logical Fiberings may – by its architecture – be added, eliminated and infinitely enumerated, thus not imposing limitations on the implementation by the mathematical representation. However, clearly, space for implementation is always limited by computational means.

Thus a feasibility study with emphasis on the database structure was conducted. The following shall give an idea of possibilities and limitations of the database usage, keeping in mind that FG uses a MySQL® software server to save the fibering data.

#### 3.8.2.1 Server Use

According to [MYS03] the most common approach of benchmarking a database are Transactions Per Second (TPS), whereas a transaction may be defined as a unit of execution a client runs against the database. This execution may be a simple select, an insert or a relatively complex command on several tables. This means that firstly, this benchmark is closely linked to the type of transaction. It may furthermore be noted that many other variables (like hardware, operating system, number of clients, database configuration etc.) may also interfere and distort results. Therefore [MYS03] suggests tests with slightly altering variables in order to obtain more confidence on results.

As a first comparison [MYS03] cites the following benchmarking of eWeek (Table 3-2):

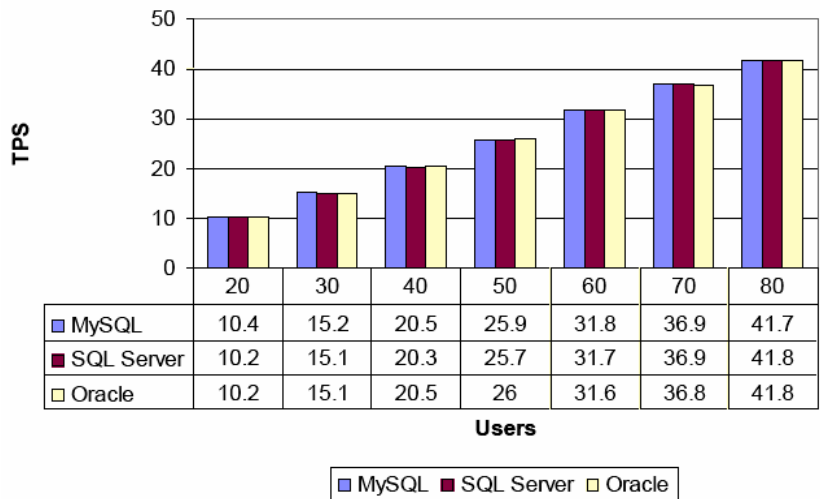
Users	MySQL	Oracle	SQL Server	DB2	Sybase ASE
100	94.735	100.700	99.724	102.382	95.753
200	186.594	199.653	181.406	203.859	191.071
300	270.824	293.318	208.900	302.783	280.535
400	361.812	384.671	205.335	398.688	370.335
500	443.559	476.241	206.676	484.065	448.888
600	523.524	544.665	204.429	322.888	475.829
700	578.082	594.806	207.359	237.747	471.294
800	599.612	615.624	200.518	215.476	471.118
900	601.788	631.388	203.829	207.247	478.718
1000	590.488	624.386	203.870	204.682	475.934

**Table 3-2: Benchmarking of five different databases**

In this case an eCommerce application was used, which issued read and write-queries to a database. The performance was measured in Web Pages per Second, which is because the “execution” of a Web page automatically caused a query to be launched to the database. Interesting interpretation landmarks are the fact that DB2® obviously did not support more than 500 users at a time very well, whereas in particular SQLServer® showed a low limit for total transactions. MySQL®, which was chosen for the implementation of FG, maintained itself at the top in this test and (in the same article) received also praise for its ease of tuning.

Another paper [SCA07] shows the results of the online Transaction Processing Performance Council on three databases, including MySQL®. Unlike the results above, it showed rather similar benchmarks for rising numbers of users and increasing TPS.

Yet other approaches (e.g. [HUB04]) examine a series of different technical aspects as platform compatibility, data types, clustering and availability etc. to arrive at yet different conclusions concerning the question, which might be the best database. Surely, the issue is not thoroughly scientific, but has also elements of marketing.



**Figure 3-15: Database benchmarking with TPC-C**

Meanwhile, by the absolute values given above, it may be deduced that in any case a MySQL® database on a server should be adequate to support the demands given by FG. This short study was maintained theoretical, as a whole series of publications exist and as FG in its first instance is to be run on a simple personal computer rather than a server. Nonetheless, it should be noted that the system is not restricted technically to PC use only.

### 3.8.2.2 PC Use

To evaluate the use of FG on a personal computer, the following test environment configuration was used:

- Hardware: Intel P4 Processor at 3 GHz, 1 GB RAM (64 MB shared memory), HD 240 GB total on two physical discs,
- Software: MySQL® 5.0 Server (database), Java® 1.6 SDK (source code of FG), Eclipse 3.1 (development framework), MySQL® Connector/J 5.0 (connection Java® to database),
- During the test auto\_commit was turned on.

Three tables were implemented exclusively for testing (see Figure 3-16). Whereas table “test1” was held simple on purpose to have a benchmark for comparison, the two further tables correspond to the data handled by FG in the following way:

- “test2” → primary data representation;
- “test3” → secondary date representation.

During the test, it was found that the differences of writing speed for these three tables was small (see Figure 3-17) and quite remarkably writing to table “test3” had the worst results. On a general basis the routine needed around 18 seconds for 1000 data sets, thus writing around 55 per second. This is undoubtedly a number much too low for a server, however, for a personal computer it might be sufficient as very seldom more than 20 incoming and outgoing connections should be executed at the same time, even under DoS.

<i>test1</i>					
<hr/>					
<b>data1</b>	int		(primary key)		

<i>test2</i>					
<hr/>					
<b>data1</b>	int	unsigned	not null	auto_increment	(primary key)
<b>data2</b>	varchar	(500)			
<b>data3</b>	varchar	(1000)			
<b>data4</b>	varchar	(20)			
<b>data5</b>	timestamp		not null		

<i>test3</i>					
<hr/>					
<b>data1</b>	int	unsigned	not null	auto_increment	(primary key)
<b>data2</b>	varchar	(500)			
<b>data3</b>	varchar	(1000)			
<b>data4</b>	timestamp		not null		

Figure 3-16: Test tables



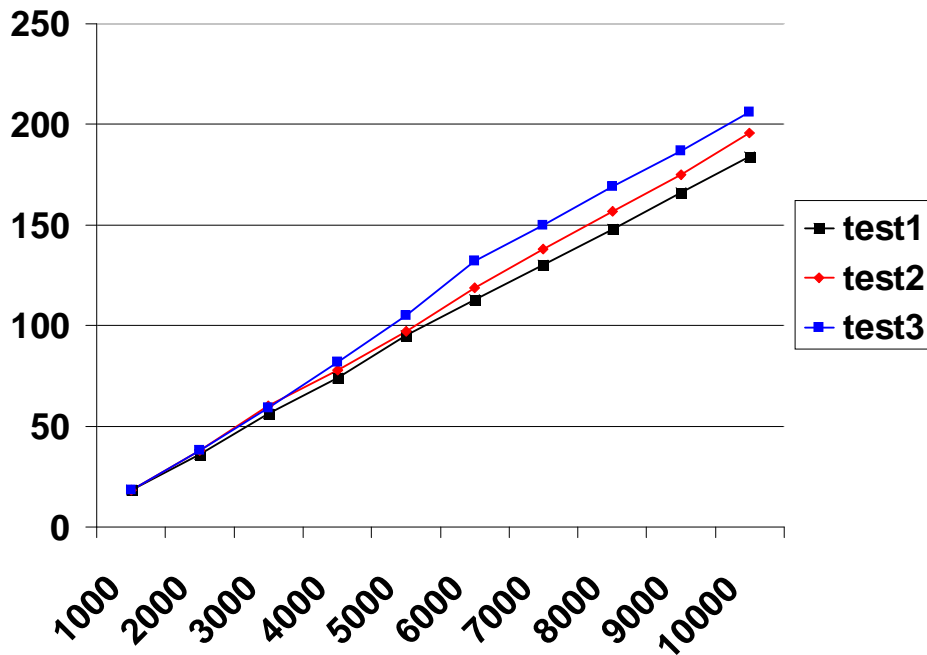


Figure 3-17: Writing speed comparison for tables “test1” through “test3”

Reading of all 10000 records by FG, on the other hand, never took more than one second. In order to estimate the impact on disc space a test with a great number of inserts was executed and the consumption of bytes on disc was calculated. During this test a total number of 1.409.294 registers was inserted on table “test3” (Figure 3-16). The difference measured on disc was of 128.581.408 bytes (ca. 128 MB), thus about 91 bytes per register. This is a reasonable number, which does not promptly put the hard disc space in risk. Only numbers over 1 billion records might be called critical as disc spaces are currently over 100 GB, already for simple PCs. It is clear however that 1 billion records also causes considerable slow-down on selects and thus database cleaning has to ensure small sets of data (more on this sub-section 3.8.4.3). It should be said that the Java® Virtual machine heap size shall be adjusted before running FG. This is true, because FG might read large data sets from the database. FG uses a maximum heap size of 128M in its standard installation, which, however, may be freely configured by the user at runtime.

To carry the whole dataset of 1.409.294 registers of table test3 mentioned beforehand into memory, FG needed 12 seconds, which results in an estimation of about 120.000 registers per second. The browsing of the records in memory meanwhile took less than 1 second.

### 3.8.3 Overview

The concrete software implementation is called Fibered Guard, which is an intelligent client firewall system for the prevention of DoS and DDoS. The system makes use of two system modes:

- Simulation: A whole series of attacks may be implemented and executed on a virtual machine. This mode is necessary to deal with one of the main shortcomings in DoS or DDoS defense system development: The lack of real-life testing possibilities [MIR04]. Standard attacks were implemented with a series of feature. For more details on attacks, see 3.8.4.4.
- Real Life: Substituting the monitoring over the virtual machine by the monitoring of the real machine, actual algorithm and methodology were maintained exactly the same, in

order to be able to benefit from the results found through the simulation algorithm. More general tests had to be executed in real life mode as attack tools are not widely available. This, however, does not diminish the given results nor their scientific validity and possible impact.

A general overview of the system architecture is given in Figure 3-18.

The system Fibered Guard is an object-oriented implementation using Java® 1.4 technology. The development has been executed, benefiting from the Eclipse 3.1 framework. The underlying database is MySQL®.

For the installation of the system the following requisites apply:

- Java® Runtime Environment (1.4 or higher)
- Pentium IV processor or equivalent
- Database MySQL®
- Reasonable disc space for the fibering data (suggested at least 2 GB)

At installation the main script has to be run against the database to create the tables and fill them with initial data (where applicable). Several attacks and attack courses have already been developed and tested, reminding that attacks are saved in the database and attack courses directly on disc.

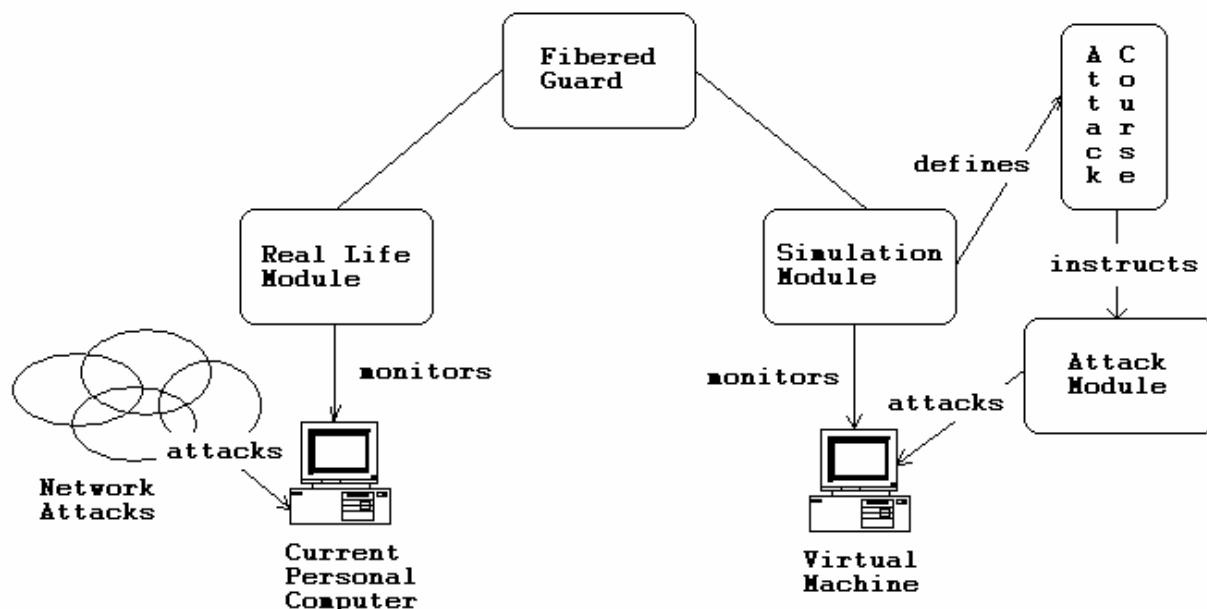


Figure 3-18: Fibered Guard Implementation Architecture

### 3.8.4 Technical Details

The following section will give an insight into some technical details of the implementation of FG, mainly to give an idea of the system's dimensions and their context. Together with the detailed description of the system's function and algorithm, which was given above, it should be a way to convey of complete picture of the project.

#### 3.8.4.1 Programming Language

FG was developed in Java®, which is known to be an object-oriented paradigm. Object oriented development styles use to structure the system by its data and not by its functions. Java®

systems are developed by a structure of classes. From each class an infinite number of objects can be created, each of which having attributes (to save and manipulate the respective data in their scope) and methods for communication and interaction with other objects and the system user [DEI04]. There is much literature on the subject, and a detailed introduction into object orientation would be clearly beyond the demands of this thesis. However, in order to justify the choice of development paradigm the following advantages, specifically interesting for the development of FG, should be mentioned [SHA97, FAY97]:

- **Reusability:** Object oriented development highly benefits from the fact that it is possible to define clearly separate units, which interact via methods. This separation enables the programmer to frequently reuse code – inside the same project and also for other projects. This was done inside FG when substituting the sensors of the test environment to the real sensors. In fact, the underlying structure was not changed; merely some pieces of the construction were substituted. This was vital for the plausibility of the simulation results and the system development philosophy as a whole.
- **Increased Quality:** Obviously, these pre-tested routines highly increased the quality of further development. The result quality and relative development speed were directly affected.
- **Mapping to Domain:** The problem domain of DoS attacks is more aptly modeled by objects than by any other paradigm as objects inside the system can represent real-world objects.
- **Client/Server Application:** Finally, Java®, through its systematic code and impositions on formatting, the system is better suited for the network environment and especially inside a security approach as reliability is crucial. Apart from that the fact that transmission is always done via methods, greatly eases the use in a networked ambience as this is its native way of communication between the nodes (packages).

The development was executed using furthermore the Eclipse framework. This is known to add many essential resources to Java® development and thus speed the development process.

### 3.8.4.2 Class Structure

The following table shall give an idea of the classes used in FG. The interplay of the classes is given in Figure 3-19. Deeper technical details are provided whenever deemed necessary to explain the main principles of the FG system. It should be noted that the following does not follow a classical system documentation style, as its main idea is giving an overview of the system’s functionality and style. More details may be found in the appendix.

Name of Class	Functionality
ANNPlugin.Java	This class contains the functionalities of the Artificial Neural Network (ANN) for validation of the Data Mining plugin functionality.
AttackCourseDefinition.Java	Class, which opens a window and implements its functionality for the definition of courses of attacks in simulation mode. Basically, a sequence of steps is programmed. At each step the connection method, the factor (number of same connections) and the delay to the next connection (which can be 0, i.e., at the same time!) is given. This definition is not saved in the database, but on disc. Before any simulation this archive of attack courses must be provided.
Configuration.Java	This class corresponds to the general configuration window and its functionalities, which leads to the specific configuration windows.

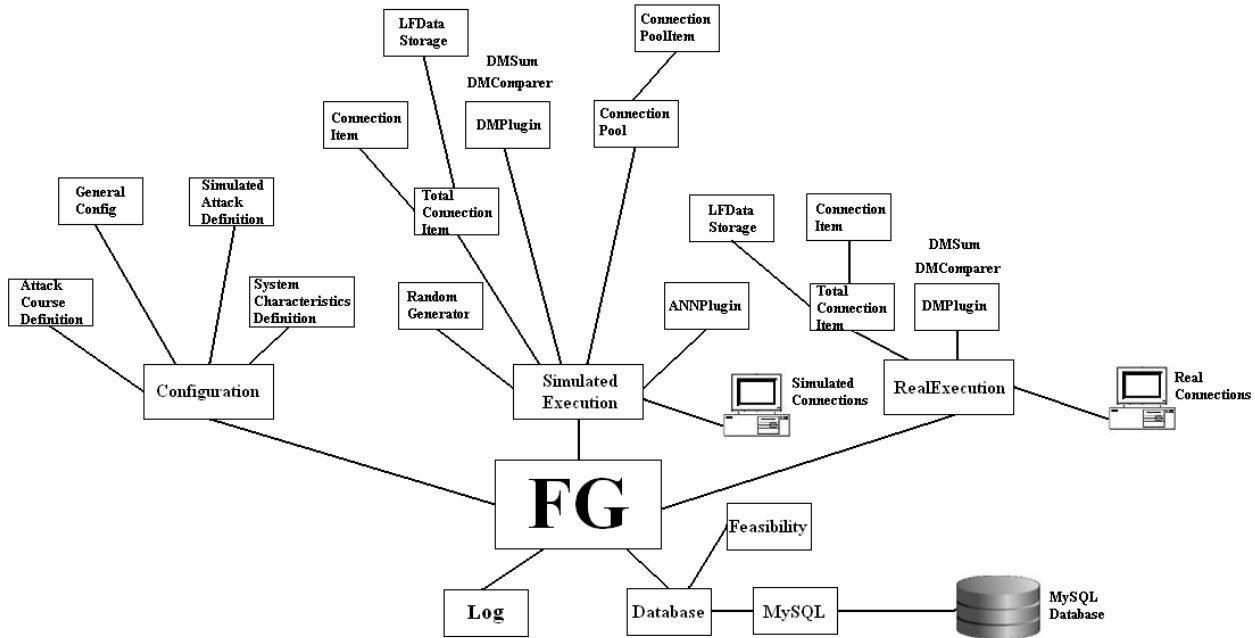
ConnectionItem.Java	Class, which holds the core data of a connection, possessing getters and setters for its definition and information extraction.
ConnectionsPool.Java	Queue, which holds the release orders as described before. FG makes use of a hardcoded limit of 10000 orders. This limit may be adjusted via a constant in the source code. It should be noted that this limit exists in the simulation mode only (i.e., there is certainly no such limit in live execution).
ConnectionPoolItem.Java	Whenever an attack is launched against the virtual machine at simulation, a release order item is added to the release-queue. The release-queue defines which item shall be released at which time with which effects. To illustrate this function, we might imagine a SYN-attack to be launched against the virtual machine. Its negative effects shall not last forever but must be released after a certain time. Thus an item is created on the release-queue and deprecated at every iteration step until it reaches its end of validation. In this case the system resources are released and the release order item is deleted from the queue.
Database.Java	Database management class with a basic framework for database usage, offering general update and query facilities. The class makes use of the connection to MySQL® using the MySQL.Java class, which establishes the specific connection with the MySQL® server. Without great impacts on development time, this may be altered to another database server, if necessary.
DMComparer.Java	Class, which implements the Data Mining comparison module.
DMPlugin.Java	Class, which implements the Data Mining plugin, establishes rule sets from the primary data representation and saves it into the secondary data representation.
DMSum.Java	Class, which is used to make the sums from the primary data representation.
Feasibility.Java	Evaluation class for database feasibility, used to obtain the data given in 3.8.2. This class serves for test reasons only and does not make part of the final FG package to be rolled out on a PC.
<b>FG.Java</b>	<b>Main class of the FG system, implementing the main menu window and its functionalities. The class establishes the connection to the database at runtime and provides it to all subclasses. Basically, the window offers the choice of the operational modes (“simulation”, “real life”) and the possibility to access the configuration window. Any configuration has to be done before the execution of either operational mode.</b>
GeneralConfig.Java	Class, which allows making general configurations both for simulation and real life execution purposes like processing delays to provide a more realistic course of simulation and logging levels, which advice the logging

	mechanism to monitor the system's activity in a specific way.
GridBagUtil2.Java	Auxiliary class used for easier administration of the GridBagLayout in Java®. This class is used for the implementation of window structures.
IndexException.Java	Exception to be thrown by the release stack in simulation when there is no item to be released.
LFDDataStorage.Java	General data type to be used with the Logical Fibering structure for storage. It holds values at the same time in several data formats, converting the incoming value automatically at the objects creation to all other data types.
Limit.Java	Class, which holds limit values for normalization speedup.
Log.Java	Class for the logging of system activity. This class was used for the automatic polling of the results given in chapter 4. The logging class is provided to every system function and can be configured to log several different events providing a "log level" in the general configuration.
MySQL.Java	Class for the physical connection with MySQL®. FG uses the MySQL® Connector 5.0.5 (JDBC) for access.
RandomGenerator.Java	Class for generation of specific random values (as random IP-addresses, program names etc.). The class is used for random generation in simulated attack courses where the user configured the specific value to be random.
RealExecution.Java	Class for the management of real life execution, modeling data storage, data retrieval and defense in the same way as the simulated execution, however, substituting the monitoring of the virtual machine with the monitoring of the real client and its connections.
SimulatedAttackDefinition.Java	Window for the definition of simulated attacks. This class works directly over the database and offers several emulated functions and possibilities of randomization to the user. The simulated attacks, which were defined with the help of this configuration window, are given in 3.8.4.4. A screen-shot of the window is given in the user manual in chapter 8.
SimulatedExecution.Java	Master class for the simulated execution. This class establishes a virtual machine with several system resources against which the predefined attack course is run. The attack course may be launched once or may be repeated. FG's defense algorithm may be run during the execution or may be deactivated in order to evaluate its impact. The virtual machine offers a specific monitoring window with specification of all resources and percentages as well as the system mode and the current iteration counter. For ease of analysis, it is possible to stop the simulation at a specific point, restart it or trace it step by step.
SystemCharacteristicsDefinition.Java	Configuration window for the definition of system characteristics to be monitored for the definition of the

	current attack situation. Each characteristic offers the possibility to define threshold values, which specify if the system is under attack or not. Also, the value of what would be considered a “jump” in resource usage is given.
TotalConnectionem.Java	Data type to offer a more complete context with all items of an attack item defined in SimulatedAttacksDefinition. Furthermore, from this class comprehends functions to create a more compact data format for an attack, i.e, a connection “signature” with respective facilities of further processing.

**Table 3-3: FG system class description**

The following diagram is to give an idea of the links between the classes used in FG. The links do not contain information of quantities (it is possible to create several objects from one class), but shows in general terms which object uses which other. Start of this diagram is the main class FG.Java instance, which accesses objects of the classes Configuration.Java, SimulatedExecution.Java, RealExecution.Java, Log.Java and Database.Java. Log- and Database-objects are created and forwarded by FG through the rest of the modules. Configuration, SimulatedExecution and RealExecution objects themselves access a series of other objects of classes. The tree structure is depicted in an intuitive manner in Figure 3-19:



**Figure 3-19: FG class interplay**

### 3.8.4.3 General Database Organization

FG’s database consists of 5 tables. The following list shall give a generic idea of the database structure and its use. A more detailed description of all tables and their attributes can be found in the appendix:

- system\_characteristics: Definition of critical system characteristics/resources to be monitored, either in simulated modes or in real life mode (in the later case only the applicable part of the data in this table is used). Threshold values may be set for slight and severe problem situations and a jump threshold can be defined.
- simulated\_attacks: Concrete definition of simulated attacks, which implements the complete structure of attacks given in 3.8.4.4. Apart from the possibility to define fix data

(e.g. attacks from a specific source), all data may furthermore be defined as random, in which case the system generates the applicable data from a random pool of possibilities. This helps in the more effective implementation of DDoS attacks, as seen in 3.8.4.4.

- `general_configurations`: All configurations, which are used for general program settings (as delays) are saved in this simple table.
- `signature_fibering`: Database implementation of the primary data representation explained beforehand. It is organized into a header and data part to better separate the visualization of shorter and longer fibers. Internal database coding is held in string values implementing the two-valued logic fibering representation.
- `conclusion_fibering`: Database implementation of the secondary data representation. Basically, this table contains the condensed rule-sets in a Logical Fibering fashion. They are later treated by the retrieval algorithm, which prepares the data for use in the defense algorithm.

#### 3.8.4.4 Simulated Connections

The following attack/non-attack connections have been configured. It should be pointed out that FG certainly models simulated attacks from the side of their effects and obtained connection data. Thus, some configurations may fit for several well-known attacks. Clearly, FG's objective is not naming of attacks, but the defense against them. The only final distinction, which is made, is "attack" or "non-attack".

Implemented connection styles:

- DoS attacks:
  - Fraggles:
    - Amplification of UDP network traffic by the times of machines on the network (which is configured by the factor in the attack course);
    - Different IP addresses, all responding to ping;
    - Short duration of connection;
    - Effects on bandwidth;
  - SYN-Flood:
    - Fast creation of long lasting ("semi-open") TCP-connections (which fall to timeout);
    - Connections from one host only;
    - Response to ping is unlikely because of IP-spoofing.
  - UDP-Flood:
    - Connection to chargen (character generator);
    - UDP connection;
    - Effect is a large bandwidth impact happening soon;
    - Connections from one host only;
    - Response to ping is unlikely.
- DDoS attacks (TFN, TFN2K, Stacheldraht, Trinoo, WinTrinoo):
  - Variation 1:
    - Protocol TCP/IP;
    - Connection from random IP addresses;
    - Answer to ping (connection from agents);
    - Effects on bandwidth and open connections;
  - Variation 2:
    - Protocol UDP;
    - Connection from random IP addresses;
    - Answer to ping (connection from agents);
    - Effects on bandwidth and open connections
  - Variation 3:
    - Protocol TCP/IP;

- Connection from random IP addresses;
    - Unlikely answer to ping (as so configured on the handlers);
    - Effects on bandwidth;
  - Variation 4:
    - Protocol UDP;
    - Connection from random IP addresses;
    - Answer to ping;
    - Effects on a series of system characteristics.
- Normal operation (“noise” to be contrasted to attack connections):
  - Variation 1:
    - Protocol TCP/IP;
    - Connection from a single host, responding to ping;
    - Small effect on bandwidth and open connections;
    - Short lasting;
  - Variation 2:
    - Protocol UDP;
    - Connection from a single host, responding to ping;
    - Small effect on bandwidth and open connections;
    - Short lasting;
  - Variation 3:
    - Protocol TCP/IP;
    - Connection from a single host, not responding to ping;
    - Small effect on bandwidth and open connections;
    - Short lasting;
  - Variation 4:
    - Protocol UDP;
    - Connection from a single host, not responding to ping;
    - Small effect on bandwidth and open connections;
    - Short lasting;

### 3.8.5 Dynamic Environment

Any project dealing with security is inserted into a life cycle of security systems [SEC07], which is a dynamic model for the continuous need of monitoring and adjustment in a dynamic security environment. Figure 3-20 depicts this life cycle.



**Figure 3-20: Model of the Security Life Cycle**

The phases of the cycle are [SEC07, APV05]:

- Assessment: This phase comprehends planning of a security model, preliminary tests, expertise about vulnerable points and the planning of the resources to be employed. This planning has been initially done in a conceptual manner, comparing the approach to



several ones present on the market and also during development by the usage of the simulation model and the obtained results.

- **Protect:** In the protect phase, a solid system architecture shall be elaborated. This claim was realized through transparency provided by the separation of system and data (Java® core and database) as well as the enforcement of best practices during development. A separate database provides the possibility of reporting with other systems. Real life protection was implemented and audited.
- **Validate:** Once in place, a security system needs to be audited to verify that it is – in fact – providing the necessary security. This validation was provided by firstly simulation tests and secondly by the test of the real-life module. Detailed evaluation may be found in Chapter 4 (Results).
- **Train:** Human operators and administrators must understand the principles of a security system in order to take appropriate action in unknown situations. FG is open to human interaction and – depending on the challenges the system is exposed to – may be adjusted. This may be done directly over the database using the intervention module discussed beforehand in 3.8.4.
- **Monitor:** Constantly the results of the system have to be monitored. This is reached by the logging facility mentioned in 3.8.4, which produces logs, which could be directly analyzed or processed by a third-party system in order to structure the information as desired.
- **Next Assessment Step:** FG is prepared for unknown situations, however, especially by the change of underlying network or hardware structures it might be necessary to adjust the system. Object oriented systems are known for their reusability and relatively straight-forward understanding, which supports the further development process [FAY97].

### 3.8.6 Limitations

The present approach is an implementation of an intelligent client firewall. Thus, the basic limitations for client firewalls apply [RAY07, SUP05]:

- A firewall cannot treat threats that come from within the trusted zone. This limits the initial scope of this project to a client firewall rather than a network firewall, as on internal networks no nodes can be really trusted concerning the suspect of DoS or DDoS,
- A firewall does not scan e-mails. DoS attacks normally do not act this way, but one might think of new technical possibilities or ideas to launch an attack. The scanning of e-mails by firewalls, in any case, is not very practical as it could slow down traffic speed considerably. For this purpose, an AntiVirus system should be used.
- As a firewall presents the entrance door to a system and all traffic must go through it, it may definitely be observed that a firewall itself could be a victim of a DoS attack. In order to prevent this problem, FG makes use of its different system modes and acts according to the attack situation (see 3.5.4).
- Specifically in the DoS attack environment, a firewall will have difficulties with the following attacks:
  - Attacks based on system bugs rather than an overload situation: The present approach may save and analyze the data, but the time window to do so may be truly narrow as these attacks may bring down a machine in small parts of a second. The best defense in this case may be a simple bug fix;
  - Physical attacks, as these do not go through any firewall.
- Successfully treated attacks are those that are based on an overload situation, which might be one of the main characteristics of classic DoS. FG focuses on this overload situation as it takes action as soon as there is an identified scarcity of a system resource.

Although being designed as a client firewall, it is merely a question of deploying the same logic to network components, if desired. From the present approach, several more advanced systems may be created, especially because the core itself is already validated.

It shall be pointed out that Logical Fiberling is a data structure which can capture an attack situation in details (as seen in 3.5.1). However, its main task is modeling, not conclusion. This is why the development of a conclusion engine was necessary. Because of the ease of handling of binary data, this conclusion engine could easily be clamped to the fiberling without any previous adjustments.

Data Mining often handles very large sets of data. This produces obvious performance problems, which also are a clear issue for future developments of the present approach. Two classical solutions are either optimization of the data or the algorithm or the use of more advanced underlying technology. A very clear claim would thus be a future optimized version in order to more efficiently use the system's resources, not losing the focus of actually not creating (or helping to create) DoS.

Most other problems are linked to the data quality itself. Sparse data may produce overfitting and reach over simplistic (or even completely wrong) rules for a much more complex environment, whereas huge and very diversified data may produce underfitting, i.e. much too vague and abstract conclusions for any useful interpretation.

These issues are currently treated by the possibility of manual administrator intervention. This intervention can be done directly over the database by the elimination of the respective datasets. Inclusions from scratch may also be possible, but are much more difficult through the Logical Fiberling data representation.

## 4 Results

The present chapter shall give results obtained with FG. In the introductory part of this chapter the general setup of the executed tests is discussed and the type of measures, which were used, is presented.

### 4.1 Overview

#### 4.1.1 Motivation

Main motivation of this chapter is the practical validation of the present approach by the measuring and the interpretation of its performance. In particular the following pre-assumptions are to be backed by the evaluation:

- FG successfully defends against DoS and DDoS attacks;
- FG presents reasonable performance compared to other approaches;
- FG especially does not block legitimate connection light-heartedly.

Furthermore, the chapter shall give an idea of the tuning of FG.

#### 4.1.2 General Method

The following results discuss simulation mode scenarios. This is due to the fact that real life DoS or DDoS scenarios do not at all provide the same power of testing as their simulations [MIR04]. It should be pointed out that a good validation of the system's principles is possible under simulation mode as far as attacks are simulated in a realistic way [MIR04]. Furthermore, the general freedom given by a simulation provides a framework of not only testing the most common cases, but also a variety of uncommon and critical circumstances at the very end of the spectrum of system limitations.

It should be stated meanwhile that FG's modular structure allows the prompt substitution of the emulated sensors and methods to real-life sensors (which were also implemented) and methods without any change in the algorithms used in simulation mode. Thus, very similar results are expected in a real-life test.

All results were obtained directly from the system by the use of a Log-file.

#### 4.1.3 Test Setup

In order to find a realistic measure of the system's performance in a problem environment, initially before every test, all fibering information was deleted. Thus the system had to find new conclusions and ways of defending in every situation. It should be said that certainly an initial set of valid information inside the fiberings provoked a more stable situation from the very beginning, i.e., the stabilization phase, which may be observed in a series of diagrams below, did not take place.

For every attack, a specific attack course was defined – a way of defining several connections for an overall period of 500 virtual machine iterations. Each attack course featured the attack itself and also contained 4 legitimate connection types with the following characteristics:

- Correct source IP addresses were provided, however, one of these connections would not respond to a ping request (as this feature may be disabled at hosts);
- Small impact on system resources, mostly on bandwidth and open connections;
- The connection did not stay in the pool for long, namely only for up to 5 iterations;

The attacks themselves had a subset of the following main features:

- Great impact amplification;

- Spoofed data and improbable (but not impossible!) response to ping;
- Long time in connection queue (15 iterations or more) for each connection;
- Multiple connections at the same time.

The more specific setup of each attack and defense method is given below together with the results.

#### **4.1.4 Measures**

The following measures were used throughout the tests:

- Overall Resource Index (ORI): This figure presents the arithmetical average of the use of all monitored resources. It is given in percent. It may be considered that many attacks do not focus on all monitored types of resources, with the obvious result of producing a low overall ORI even under a severe DoS situation. It is, however, the most adequate index to analyze the overall system's health state and its development as it considers all resources. As it is certainly difficult to establish degrees of importance for each resource (e.g., is bandwidth more important for the creation of DoS than disc space?), for the time being, all resources are considered of exactly equal importance in the ORI.
- Critical Resource Index (CRI): This index provides the state of the currently most critical system resource in percentage of use. It was mainly created to give an idea of the present threat on the system. Unlike with the ORI, this index does not consider the overall situation, thus not taking into account any system resource development below the measure of the critical resource.
- Legitimate Blocked Connections (LBC): This is a highly critical measure as an adequate system for DoS and DDoS defense must give great value to legitimate connections. Not taking this measure into account is a gross conceptual mistake as DoS and DDoS attacks aim at firstly the denial of an offered service on the network and only secondly – as a side product – at bringing hosts down (the later has the clear advantage of guaranteeing the DoS situation further, thus it might be seen as a second, but not a principle aim). An example of approaches, which act blindly upon DoS and DDoS are especially congestion control systems. The unsatisfactory result of such an approach is given in 4.3.2 below.
- Illegitimate Blocked Connections (IBC): Whereas legitimate connections shall be maintained at a maximum level, illegitimate connections, i.e. attacks, must be blocked. This successful distinction is one of the main merits of FG and demonstrated below.
- Illegitimate Connections (IC): Total number of attacks.
- Legitimate Connections (LC): Total number of legitimate connections (non-attacks).

#### **4.1.5 System Parameters**

FG may be tuned with a series of parameters. A list and explanation is given in the appendix. Concerning some parameters further explanations are given below, as their change produced interesting results, which may guide the way to a correct tuning of the system or future implementations.

### **4.2 Defense against Attacks**

FG's main purpose is the defense against DoS and DDoS attacks, meaning that attack connections shall be blocked and legitimate connections shall be left unharmed as good as possible. Thus, a series of attacks was analyzed.

## 4.2.1 DoS Attacks

DoS attacks are basically attacks launched from one or a smaller universe of attack sources. The evaluation of two of these attacks is given in the following. It should be pointed out that several others tests were undertaken and the following results are typical examples of different types of DoS attacks.

### 4.2.1.1 Fraggle

The Fraggle attack scenario was implemented with the following characteristics in FG:

- Connections from several IP-addresses;
- All of these addresses are valid and answer to ping;
- High effects on bandwidth due to a great series of attacks at the same time (impact equal to 40 attack connections in FG, which corresponds to the number of nodes on a small company network);
- Impacts on network bandwidth.

Firstly, we examine the success FG obtains with the blocking of the attack connections (see Figure 4-1). The figure gives the accumulated number of connection attempts (y-axis) and the number of iterations (x-axis), examining, how many attempts were made (blue line) and how many of these attempt were blocked (red line). It may be observed that there is an initial tradeoff due to the fact that no fibering information is encountered in the database from start. However, after around 50 iterations, the threats are recognized and treated. It should be noted that FG only treats attacks, which have the potential to be successful, i.e, which in fact produce problems. Therefore, as a start of the processing, it is always necessary that a slight or severe problem occurs. It should be noted that – out of this reason – some tested attack configurations fell flat as they did not really produce the expected problems.

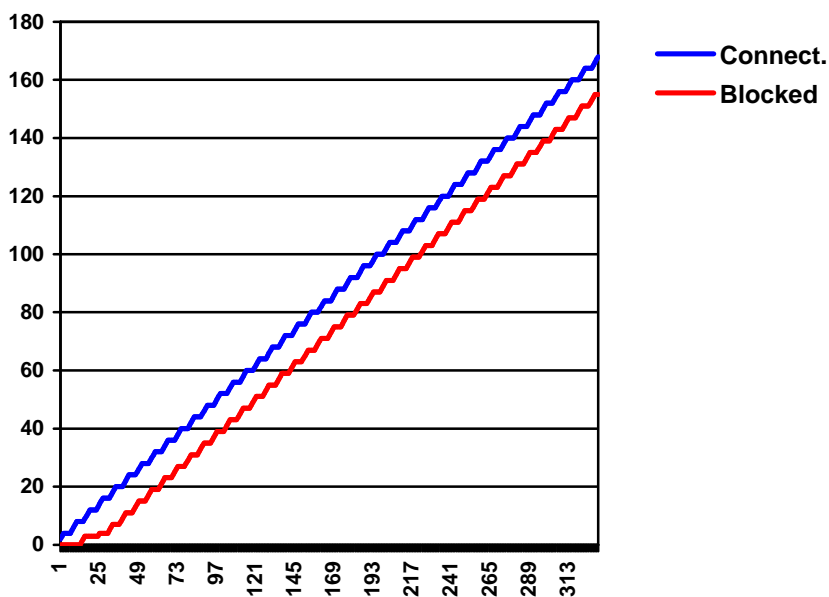
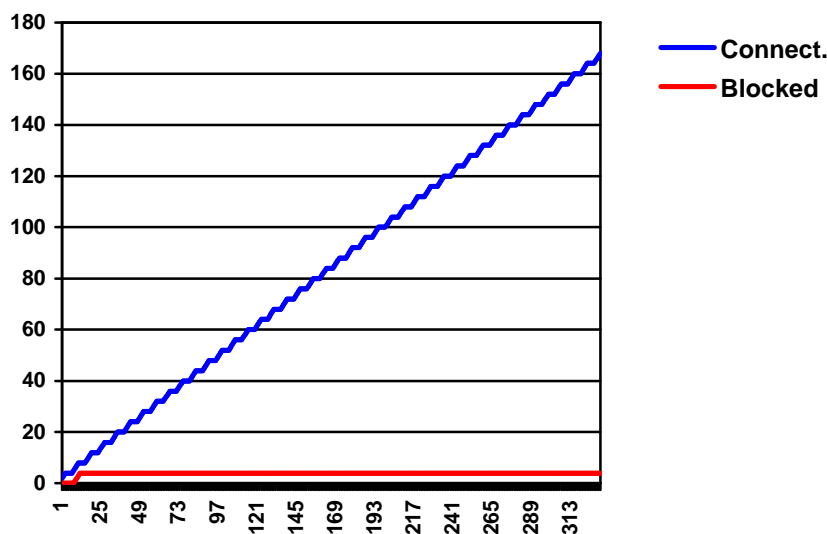


Figure 4-1: Fraggle – treatment of illegitimate connections

Secondly, the treatment of the legitimate connections was analyzed (see Figure 4-2). Again, we have a look at the accumulated overall connection attempts (blue line) and the accumulated number of blocked connection attempts (red line). It may be observed that a small number of connections was initially blocked. This is mainly due to the random blocking behavior in the

severe problem phase and not because of incorrect Data Mining assumptions. The later is proven by the fact that there are no additional different blockings after the initial phase. Although this type of behavior was thoroughly typical during any DoS attack processing (see, e.g., Figure 4-5), it must be said that in a smaller ratio of cases (15 of 100 executed tests with the present attack scheme), the initial conclusion finding phase was longer, extending itself over 200 iterations. This happened mainly, because of very noisy data (the first attack connections were saved as normal connections due to the fact that no problem had yet occurred and the legitimate connection attempts were made right upon the problem situation). However, as the system learns over time, defense was correctly established around iteration 250. Furthermore, it should be said that in none of the DoS attack tests a total or majority block of legitimate connections occurred. This way, it may be concluded that FG recognized the threats and legitimate connections successfully and treated the situation correctly, in a fully automated way. As we will see later on (in 4.3.1) no manual approach is ever prepared to reach such defense goals.



**Figure 4-2: Fraggile Attack - treatment of legitimate connections**

Meanwhile, correct treatment of connections is not enough. At the same time, it is necessary to have a look at the overall health state of the system as even few attacks may have devastating consequences and a defense, which comes too late, practically fails on the whole.

As explained above (4.1.4), we mainly examine two indexes: The Overall Resource Index (hereafter abbreviated as ORI), i.e. the average of all used resources and the Critical Resource Index (hereafter abbreviated as CRI), which evaluates the resource with the worst performance (which is thus the most critical for the creation of a DoS situation).

Again, it must be said, that FG depends on a problem situation to happen as it uses exactly this situation for its conclusions. The ORI diagram (given in Figure 4-3) shows this initial problem situation with its total impact of around 16% of all system resources (y-axis: used system resources, x-axis: iterations). With the establishment of the correct conclusions around iteration 50, the used system resources fall to a very low level, which corresponds to the impact of merely the legitimate connections.

This way, FG's positive effect on the system resources use in a typical Fraggile DoS attack situation was proven.

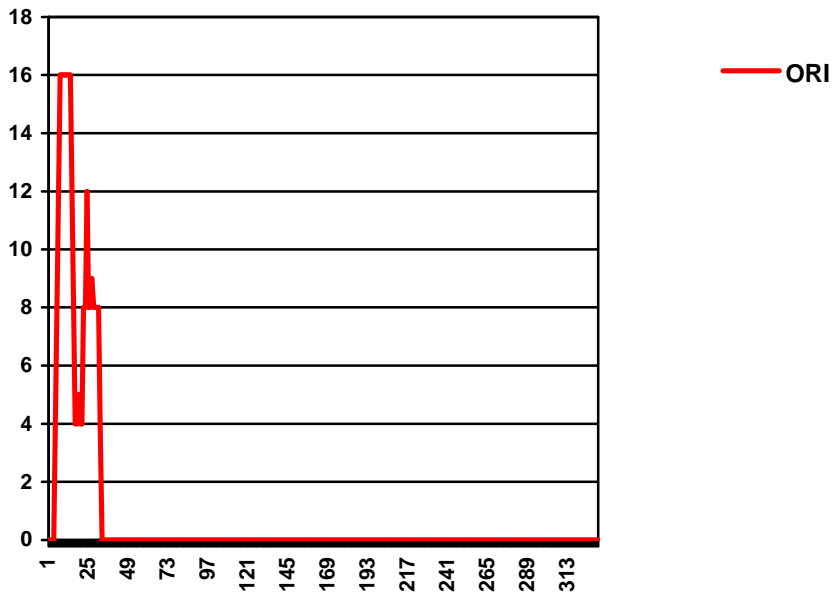


Figure 4-3: Fragge: ORI

Finally, it should be interesting to look at the CRI in order to gain an idea of how critical the situation actually got during processing. The following diagram (Figure 4-4) gives firstly an idea of this critical resource situation and how it was treated. As FG orients itself mainly by the critical resource situation, the CRI gives a good insight into the overall processing. Secondly, in order to better demonstrate the system's merit, we contrast the CRI with a situation that offers no defense whatsoever.

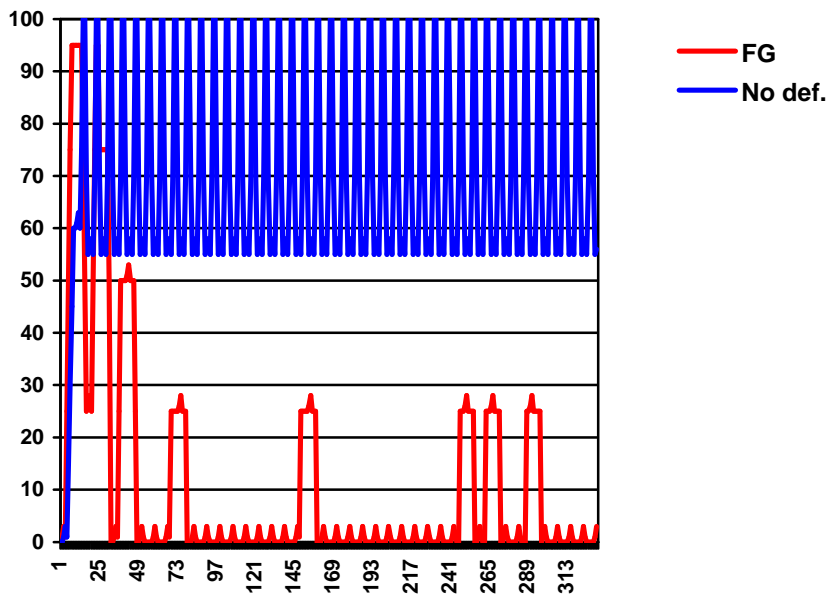


Figure 4-4: Fraggle: CRI

In Figure 4-4 (x-axis percentage of resource usage and y-axis number of iterations) we clearly see a great positive impact through the use of FG. Certainly, at 95% of system resource usage, we have a severe problem situation, which leads to random blockings to survive. As this blocking is, however, accompanied by intelligent analysis it is only a short event with little impact on the overall situation of legitimate connections.

Meanwhile the situation without the use of any defense algorithm is clearly extremely critical as we are actually passing the boundary of 100% resource usage (i.e., DoS!). In FG simulation it was assumed that a system may always “come back” after DoS. However, it should be stressed that in practice, this is not always true. Thus, the situation without defense is highly risky for the system operation as a whole and must at all costs be avoided. The merit of FG in this case is clear.

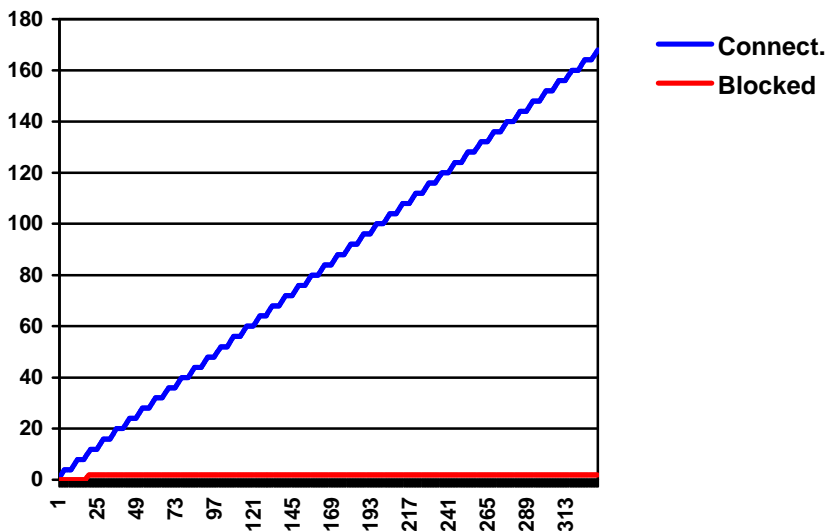
#### 4.2.1.2 UDP Flood

In the following we discuss the effects of FG under another DoS attack, namely the UDP Flood, using the same diagrams as in 4.2.1.1. This is mainly done in order to demonstrate a phenomenon which was observed throughout several different attacks: The variations of FG’s overall system performance were minimal. This, indeed, is an interesting fact as DoS attacks come with a whole series of different features and it may be suspected that many yet unknown forms of attacks are still to come. Thus, flexibility is very important. It should be yet again stated that FG is a general framework and does not have specific routines for specific attacks.

In the following result diagrams, we use UDP Flood attack streams with these characteristics:

- Protocol UDP only;
- Source IP given, but not responding to ping (spoofed);
- Program: connection to “chargen” (character generator) – see explanation in chapter 2;
- Effects on network bandwidth;
- Attack connections staying in for 20 iterations per connection.

The following diagram gives an overview of the treatment of legitimate connections:

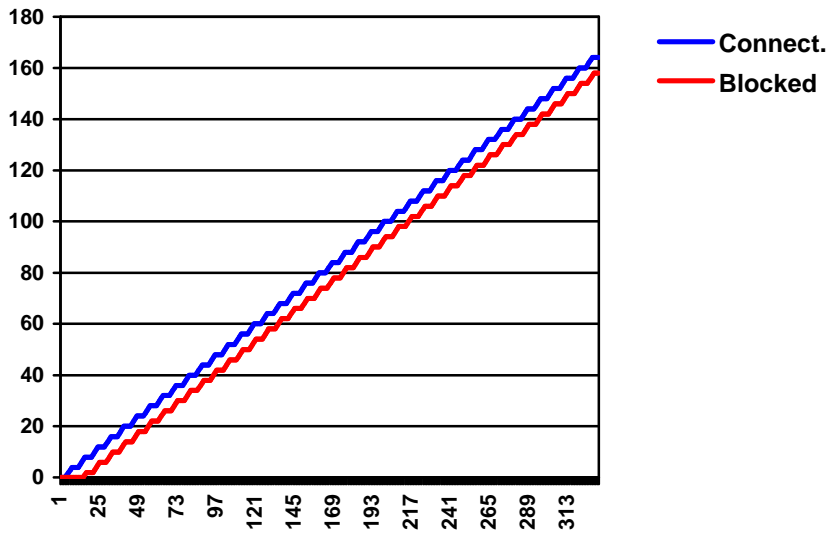


**Figure 4-5: UDP Flood: Legitimate Connections**

Comparing it to Figure 4-1, it may indeed be said that the same conclusions can be drawn from both: A very small part of legitimate connections is dropped throughout the whole course of the attack.

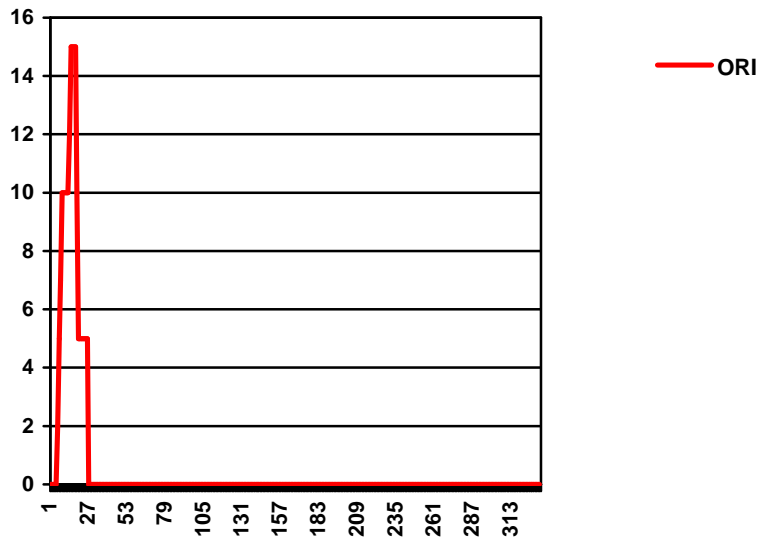


As the attack situation was correctly perceived around iteration 30, the result given in Figure 4-6 (below) may appear straight-forward. Sooner yet all incoming illegitimate connection are identified and blocked. Both graphs (blue=all incoming connections, red=blocked connections) run almost parallel from iteration 25 onwards. As a matter of fact, this repeats the results given above, whereas they are yet slightly better. Over the analysis of several attack courses (100 tests per attack type), it may, however, be stressed, that this improvement is not very characteristic (it happened in around 20% of the cases) and could be traced back to the overall attack intensity (which was slightly different – see Figure 4-8).



**Figure 4-6: UDP Flood: Illegitimate Connections**

Finally we shall also give an idea on the system resource use, as above. Figure 4-7 shows the development of the ORI upon the simulated UDP Flood attack:



**Figure 4-7: UDP Flood: ORI**

It shall be noted that the ORI is a little lower in the initial phase than in the case of the simulated Fraggle attack (Figure 4-3), which explains the very slightly different defense success.

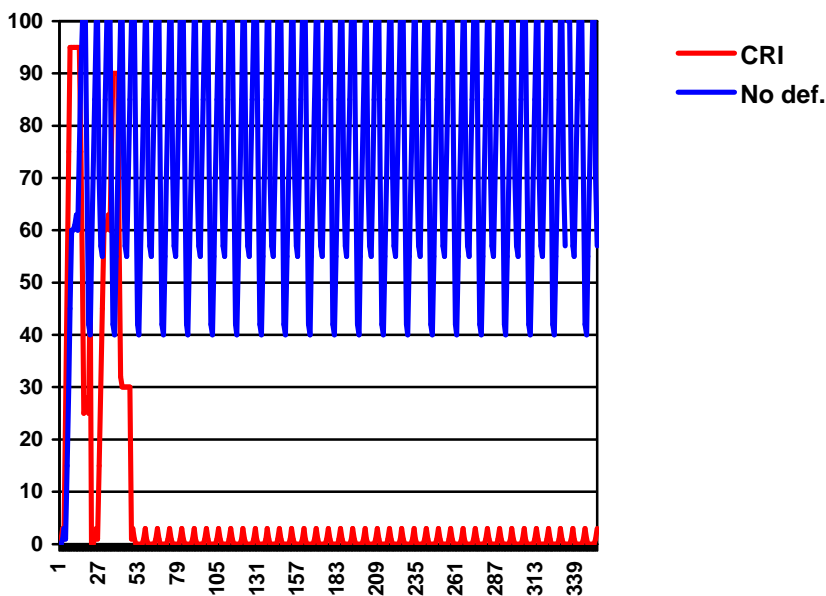


Figure 4-8: UDP Flood: CRI

Meanwhile, Figure 4-8 demonstrates FG’s very positive impact on the overall CRI situation: After the initial use of up to 95% under the use of FG, the CRI drops to a very flat level, without any further peaks (as shown in Figure 4-4). Yet again, no defense would have disastrous effects, potentially bringing the system down with peaks of 100% (and the respective “automatic” DoS connection drops, which result).

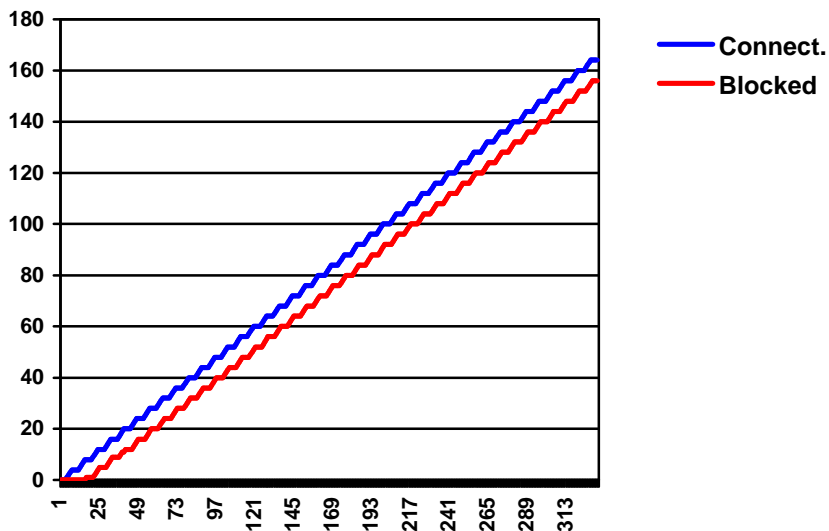
It may be said that FG showed very successful behavior against DoS attacks, which was proven by the detailed analysis of two typical examples.

### 4.2.2 DDoS Attacks

DDoS attacks make any defense system face additional difficulties, mainly through apparently random IP information, random effects and high random impacts. In FG’s simulation, we made use of several DDoS attack emulations. These have the following characteristics:

- Random Source IP;
- Random protocol;
- Improbable answer to ping (spoofed information);
- Random Source DNS information;
- Random impacts, mainly on open connections and bandwidth;
- Long attack periods of around 50 iterations per connection.

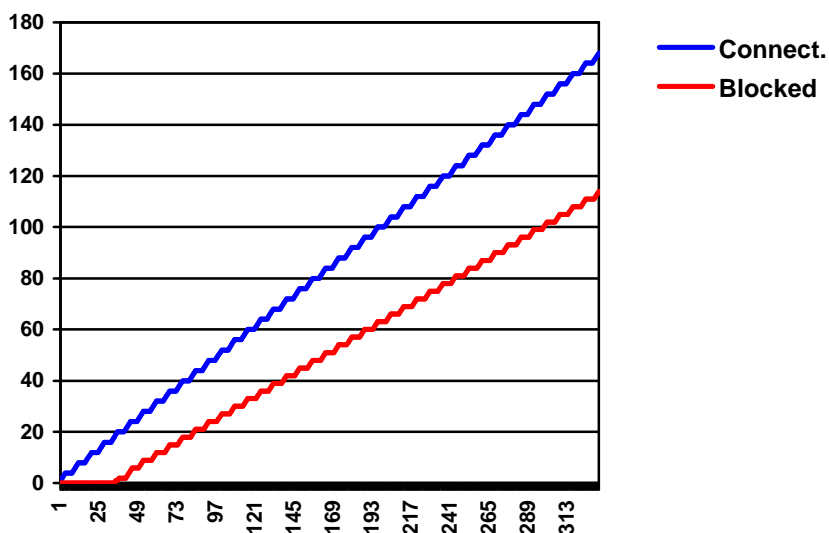
Firstly, we analyze the success of FG in blocking attack connections (see Figure 4-9). As in the diagrams above, the y-axis gives the accumulated number of connections and the x-axis the number of iterations. The universe of all accumulated connection attempts until a certain iteration is given by the blue graph and the number of blocked connections by the red graph. Remarkably, Figure 4-9 appears very similar to Figure 4-6. Indeed, attack connections are very successfully found and blocked, even in a scenario with a lot of spoofed and random information and possibly high (and fast) impacts on system resources.



**Figure 4-9: DDoS: Illegitimate Connections**

Meanwhile, it should be said, that – unlike in the DoS attack situation courses given above – there were higher difficulties in sorting out legitimate connections, which is depicted by Figure 4-10. In fact, a high and constant percentage is blocked as the initial conclusions were made with a great amount of noise produced by spoofed DDoS data.

A possible solution to this problem might be a reduction in the validity time of rules (see Figure 4-20), so that new conclusions may be found soon and thus new chances of more plausible processing may be undertaken. However, it must be said that the present results could be interpreted as still very interesting, as all DDoS threats are successfully blocked after an initial phase and a reasonable ratio of legitimate connections gets through, yet under the present attack.



**Figure 4-10: DDoS: Legitimate Connections**

The random impacts, mentioned beforehand, may be noticed in Figure 4-11: In fact the CRI development shows random peaks in the “no defense” graph, yet again most of these peaks reaching DoS situation and the corresponding risks (mentioned above). Still, the graph shows that in this attack course high system resource consumption was more constant than in the DoS situations given above (observing that the lowest CRI was around 70% after the initial phase). An interesting fact is that after a longer initial settling phase (up to iteration 115), FG manages to keep the CRI at a very constant low level (less than 5%).

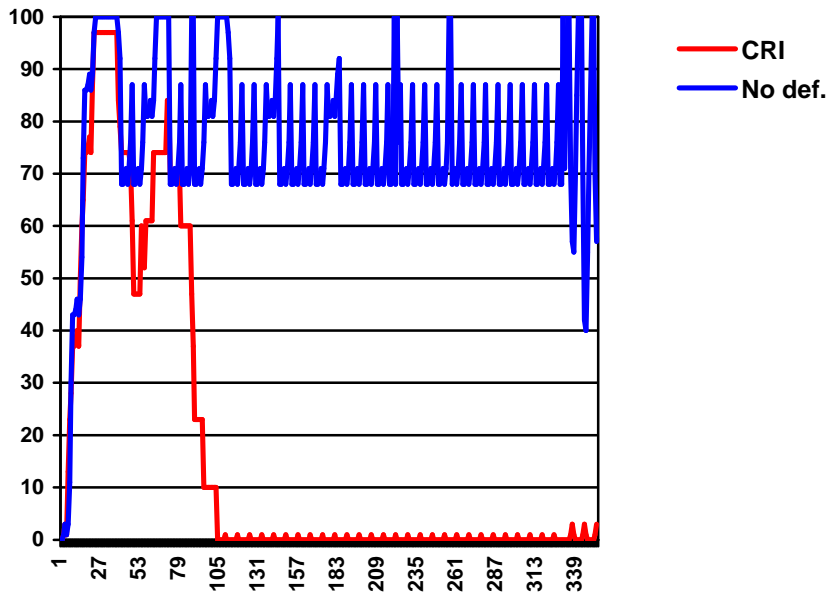


Figure 4-11: DDoS: CRI

We may extract from these results that FG successfully managed to protect the system it was running on, from the imminent DDoS attack. A reasonable ratio of – almost only – legitimate connections got through. Considering that real DDoS attacks might not provide completely random data (as given in the simulation), it could be expected that FG performed yet better in a real environment.

### 4.3 Comparison with Other Approaches

In the following part we are giving a projection of the behavior of two very common and simple approaches. Motivation of this comparison is a yet deeper insight into the benefits of FG.

At each evaluation, the method of projection is discussed. Whereas a conceptual comparison between FG and a large range of different defense methods is given in chapter 3, in this part merely two are evaluated as data on other systems may be difficult to obtain (mainly concerning the intelligent approaches), which is especially true as any coherent test must be made in a standardized test environment (i.e., the respective algorithms would have to be ready for use in the FG test environment).

#### 4.3.1 Straight Forward Approach

As discussed in chapter 2, a very common approach is that of “general measures”. In fact, enterprises that follow this approach may encounter serious difficulties if an attack finds “the gap” in the protection measures and gets through to the system resources. In this case the only apparent chance of defense would be an analysis of the log files and manual blockings of

ranges. Basically, if one does not want to block simply everything, some rules may be set in place with the obvious effect of not finding the correct blocking universe. Result may be a situation as shown in Figure 4-12. The straight-forward approach is implmented via blockings of randomly found connections. The impact on the overall situation (displayed by sporadic “down-peaks”) is indeed minimal.

In fact, any manual or over-simplistic way of DoS treatment only results in a yet greater mess, as illustrated in [MIR04].

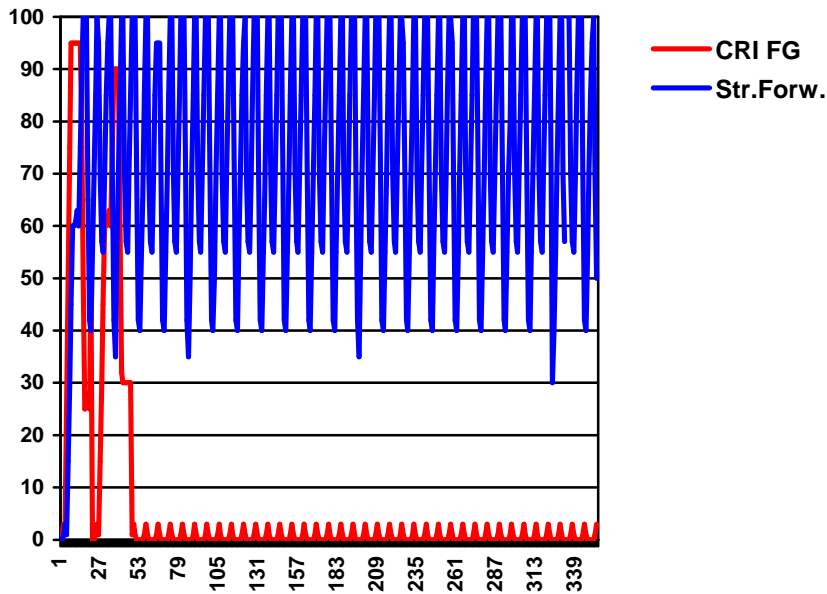


Figure 4-12: Comparison with Straight-Forward projection (UDP-Flood)

### 4.3.2 Congestion Control Algorithm

Another very common way to “effectively” treat DoS and DDoS attacks are congestion control algorithms. These employ a series of rules to systematically (or randomly) drop connections and free space. The impact is easily guessed, but its extent shall be yet better illustrated by the results in Figure 4-13. Considering that during around 320 iterations (x-axis), a total universe of 160 connection attempts were made, and we use a random blocking algorithm, both attacks and legitimate connections will be blocked by the random percentage (in the case of Figure 4-13 we fixed 50%). Result of this is a questionable positive impact on the system’s resources, whereas there is truly no benefit in the terms of DoS as – in fact – we defend against any type of connection. Supposing that we do not have equal universes of attacks and legitimate connections (attacks may highly outnumber legitimate connections), this algorithm produces yet more alarming results by total figures.

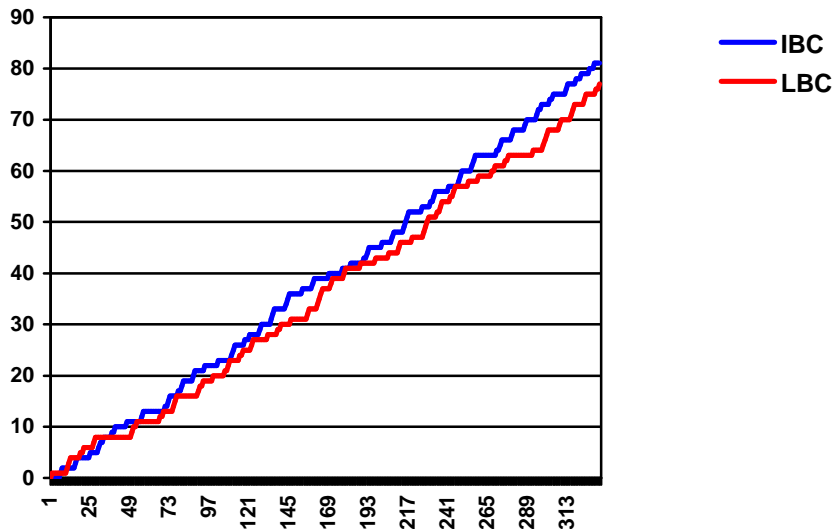


Figure 4-13: Random Congestion Control (50% blocking) with Fraggles

## 4.4 Validation with Artificial Neural Network

As outlined beforehand, an Artificial Neural Network has been implemented in order to validate the results obtained by the Data Mining algorithm. The Artificial Neural Network consists of a Perceptron, which learns the data of the Primary Data Representation and the resulting situation of each register and makes predictions afterwards about new incoming connections.

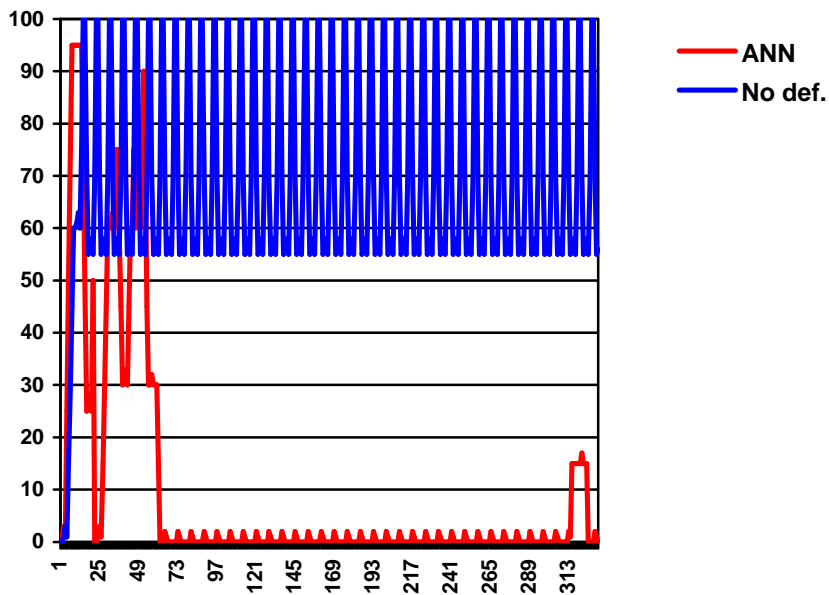
The internal configuration used for the tests below was:

- 15 Hidden Layer Neurons;
- 10 Learning cycles of all database registers through Backpropagation, triggered by every new database register's inclusion;
- Learning rate of 0.35.

Firstly, the overall performance from the point of view of CRI shall be given. Considering the evaluation of a Fraggles attack scenario in Figure 4-14, it should be said that the ANN processing does have equivalent effects on the saving of system resources, when comparing it with the DM Analysis. It should be noted, certainly, that the general function of the FG algorithm was maintained untouched and only the analysis plugin was substituted. Therefore, the fact of not letting the attack strike through to DoS around iteration 25 can be traced back to the urgent defense method of randomly dropping connections under severe problem operation – a method, which comes from the system's framework rather than the ANN.

It should be considered that an ANN initially appeared very interesting for the analysis of DoS attack situations out of basic ANN characteristics [HAY01]:

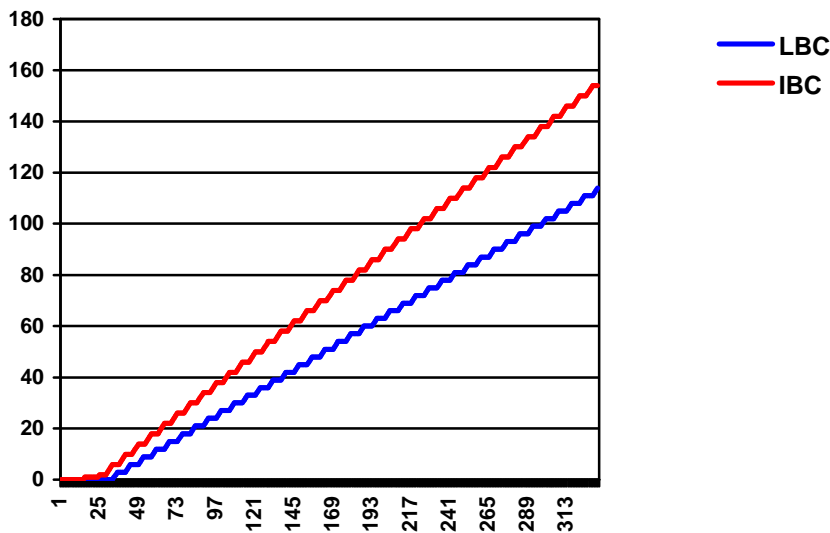
- Adaptation to noisy data;
- Adaptation to unknown data.



**Figure 4-14: Fraggle: CRI (ANN)**

However, considering Figure 4-15, a high difference between the treatment of legitimate attacks by the DM algorithm and the ANN algorithm may be noted. As a matter of fact, the ANN yet manages to block a high ratio of all attack connections (reaching 100% in higher iterations). The only problem presented is the fact that legitimate connections are to a considerable percentage interpreted as if they were threats, and blocked as well.

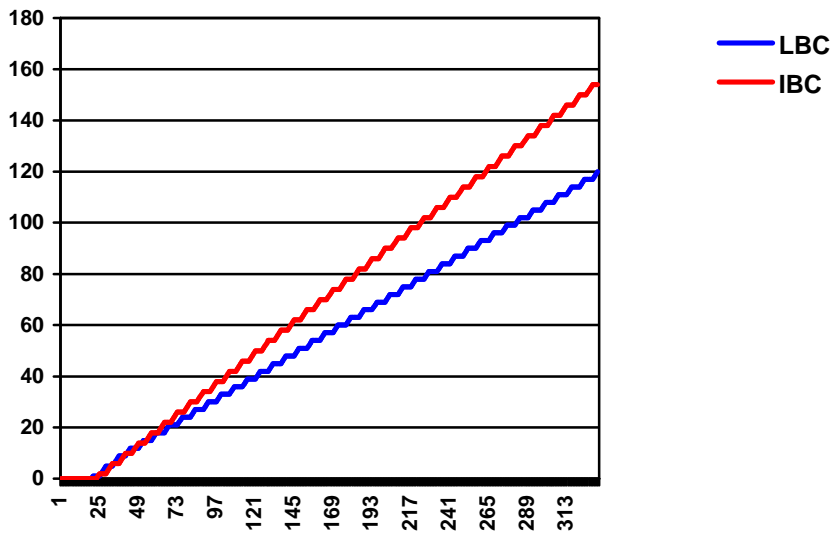
This subject might be treated by a further pre-processing of the inputs before presenting them to the ANN (as done in [SCR03]). We will not go further into this here.



**Figure 4-15: ANN with Fraggle: Blocked Connections**

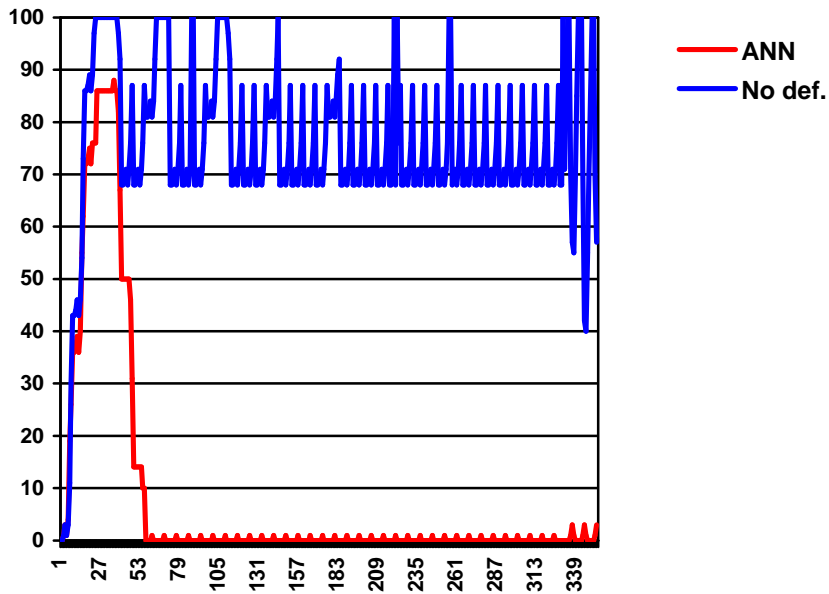
Analyzing Figure 4-16, it could be said that – at least – DDoS treatment results do not differ too much from simple DoS treatment results when using an ANN. Meanwhile, concerning the DM algorithm, some differences of the treatment of legitimate connections were quite apparent (see

Figure 4-2 and Figure 4-5). These are not found contrasting Figure 4-15 and Figure 4-16 which means that the ANN should a more sturdy way of processing, with, however, clear disadvantages over the DM algorithm concerning the treatment of legitimate connections in simple DoS situations.



**Figure 4-16: ANN with DDoS: Blocked Connections**

Finally, the CRI analysis (Figure 4-17) shows a slightly better initial phase of the ANN with only up to 85% of critical resource usage during the first iterations. It must be remembered, though, that this is at the cost of some legitimate connections and thus is a questionable advantage.



**Figure 4-17: DDoS: CRI**



## 4.5 Tuning

FG makes use of a large series of system parameters. Whereas most of them were easily adjusted by a few tests and did not really show interesting results during variations, some selected examples of more significant parameters are given in the following to convey an idea of the correct tuning of the system and the parameter's influence on the overall system activity.

### 4.5.1 DM Analysis

In its standard version, FG makes use of a pre-configured DM Analysis ratio of 50%. This means that during 50% of all cases when a new connection is established, Data Mining analysis is carried out on the database (i.e. a blocking and non-blocking rule are extracted from the Primary Data Structure). This has obvious advantages as Figure 4-18 shows:

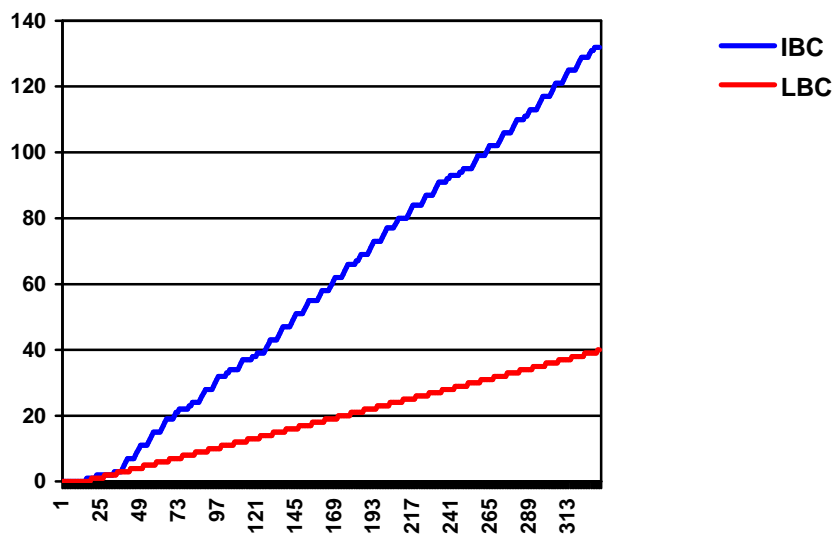


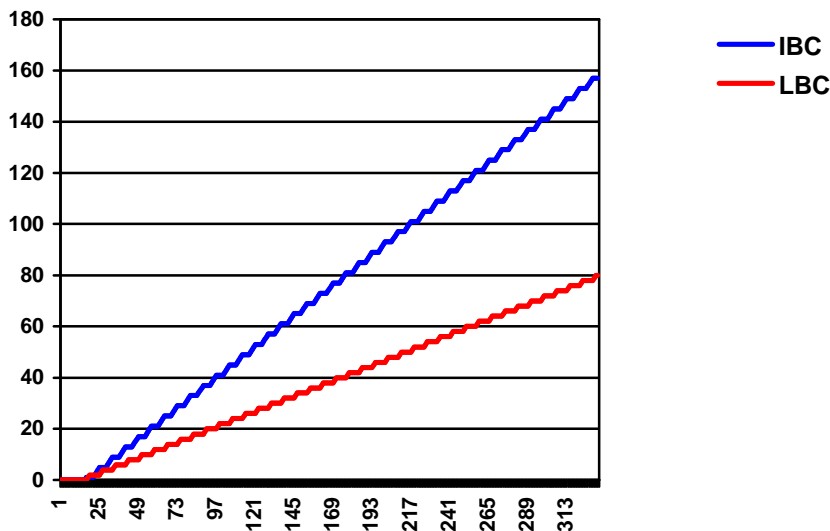
Figure 4-18: DM Analysis Ratio of 70% and Fraggle Defense

The risk of wrong conclusions on legitimate connections (as seen by the LBC above) moved to more than 50% of all cases, which is quite remarkable as the analysis rate had only been adjusted from 50 to 70%. Truly, the Data Mining analysis needs a delicate balance of parameters, considering that too much analysis in an inconvenient spot of time may produce overfitting results (see chapter 2) and fatally lead to a wrong treatment of legitimate connections (or even attacks).

### 4.5.2 Rule Precision

The same phenomenon could be observed concerning the rule precision parameter. The standard tolerance for non-fitting data sets is 75% in FG, i.e.,  $\frac{1}{4}$  at most of all sets may contradict the found rule.

Experimentally, this number was elevated to 90%, i.e., only 10% of all sets could be contradicting to the rule established. The impact was reasonable and is shown in Figure 4-19:



**Figure 4-19: Rule Precision of 90% and Fraggile Defense**

LBC rose from very few total connections (see Figure 4-2) to around 50%. This means that this variable probably needs to be adjusted empirically in real-life execution, according to the circumstances – in case of noisy environments a higher tolerance for imprecise rules is suggested.

### 4.5.3 Rule Validity

As connection situations are dynamic, constantly changing their overall scope, rules have to be constantly updated. This is periodically done by the main algorithm in the event of the following:

- The produced rules fail the minimum configured number of discrete elements, meaning that the rule is dangerously general and could block great quantities of legitimate connections;
- The rule validity is reached. This rule validity is applied by constant database checks and was elevated to two minutes in the above cases, which means, that it was – in fact – not reached during operation.

Therefore the evaluation of the effects of rule expiration had to be studied, mainly as the event is clearly necessary, but the effects could, in theory, have produced random conditions.

In the following test, the rule validity was defined with 3 seconds only. In the below case this means that rules kept expiring throughout the process. The whole simulation was, thus, on purpose kept at over 1 minute by the means of configured delays.

Considering Figure 4-20, we might say that instabilities by this rule change are minimal. The only visible impact is on the total number of blocked attacks (125 compared with 165 in Figure 4-1), whereas the number of blocked legitimate connections stayed very low.

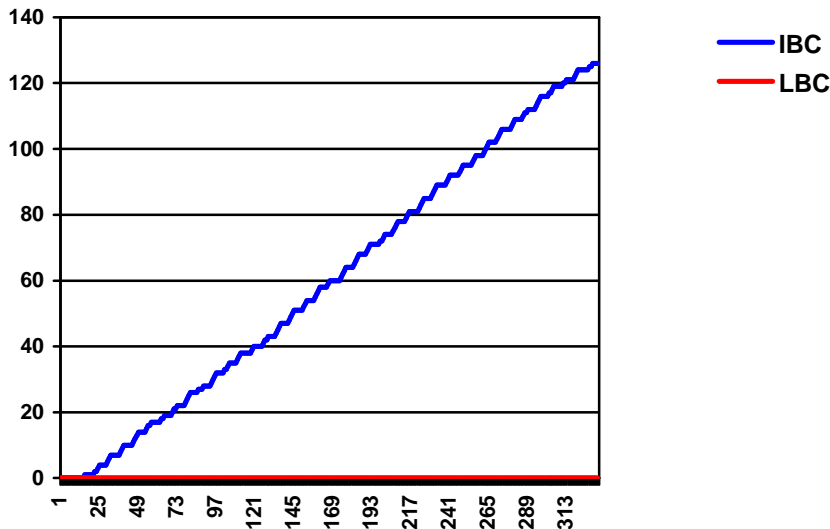


Figure 4-20: Rule Validity of 3 seconds and Fraggle Defense

#### 4.5.4 Save Rate

Finally, at normal operation, with a ratio of 80% (this means, at 80% of all connect attempts), we are saving data into the Primary Data Structure with the motivation of using it in later problem conditions as “contrast”. As this is resource consuming, the parameter had to be justified. Thus, it was lowered to 50% and the effects were studied. It may be said that no changes whatsoever happened concerning legitimate connections, but a very small tradeoff (around 5 connections) of attack treatment could be noted.

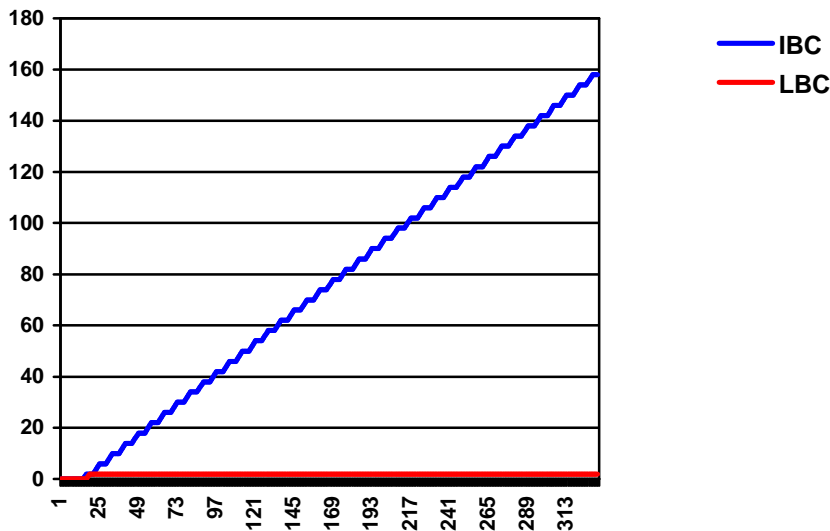


Figure 4-21: Save Rate of 50% under normal operation and UDP Flood defense

In general, it should be evaluated if such a small part is significant for defense (which could be the case in huge attack scenarios). If not, the ratio must be kept low to ensure light-weight processing, one of the main principles of FG.

## 5 Conclusion

In this thesis an intelligent generic approach for the defense against Denial of Service and Distributed Denial of Service attacks was elaborated.

In a first step, a thorough examination of the characteristics of DoS and DDoS attacks showed that these were mainly based on spoofing and the creation of an overload situation while generally using openly available entrance doors to systems. The overload situation in this case has the primary goal to make a system stop to respond on a network, which is often misinterpreted as the intention to “bring down the system”.

Secondly, a series of defense mechanisms was studied. None of them could be considered really satisfactory, because of several reasons:

- Being over-simplistic;
- Concentrating on the host’s health state and not legitimate connections;
- Not managing to reach reasonable conclusions;
- Misinterpretation of the very nature of DoS/DDoS.

With the motivation of offering an alternative to the current approaches, the system Fibered Guard (FG) was developed, based on the following areas of research:

- Logical Fibered Guard for data representation;
- Data Mining, specifically a classification rules algorithm, for rule extraction;
- Artificial Neural Network for validation;

Fibered Guard respects the following main principles:

- Honoring of legitimate connections;
- Light-weight operation.

Whereas these points may appear straight-forward (as a contradiction to them helps or creates DoS), it was observed that they do not seem to be known to a variety of existing defense systems.

Fibered Guard showed good results against DoS and DDoS alike, which was proven by the evaluation of a series of tests in chapter 4.

The present conclusion shall give a summary of achieved benefits and known limitations, leading the way naturally to future development issues.

### 5.1 Achieved Benefits

The following benefits were achieved with the development of FG:

- Good defense against several DoS and DDoS attacks: This has been proven by the results given in chapter 4: Legitimate connections were mostly left untouched, the system’s health state was preserved and attacks were blocked. A special characteristic in this case is the distinction between legitimate and illegitimate connections in an automatic and reasonably accurate way: The guiding line used in this analysis is the monitored health state of the host system, i.e. an empirical figure. The noise produced during analysis is successfully treated by the implemented algorithm, as shown in chapter 4.
- Adequate system operation: Many systems fall into deep analysis at times, which may be considered at least unnecessary, or even inconvenient. It should be kept in mind that a DoS attack actually aims at denying a service. This is mainly done by overload situations. The analysis itself might – if not correctly used – be abused by attackers to elevate the overload. This is why FG adjusts itself to the system’s health state, only analyzing and defending in problem situations.
- Modular structure: The present system is meant as a step in development. Several intelligent (or non-intelligent) algorithms may be applied by simple means and the environment sensors may be changed from emulated simulation sensors to real-life sensors, thus preserving fully the system’s core.

- Complete testing environment: FG makes use of a virtual machine, with the possibility of connection establishment and the administration of critical system resources. This machine was used for the successful validation of the system's principles.
- Automated operation: FG's main operation is automatic. Although – as in any intelligent system – it is necessary to offer means of human intervention, it can be said, that only an automatic approach survives the high load produced by DoS attacks (see chapter 4). It was, indeed, a mandatory system feature.
- Expressiveness of rules: Through the Logical Fibering structure the condensed saved data reached an additional connectivity by logical connectives, thus giving it a great power of expressiveness.
- Reduction of situations to its primitives: Whereas many systems treat incoming information only on their explicit (and somewhat abstract) values, FG reduces the current situation to a binary representation and works with its primitives. This gives the intelligent algorithm the possibility to “see” connections, which other algorithms will certainly pass by. Out of the often “tricky” environment of DoS attacks, this is actually a more adequate way of processing.
- Readability of rules: Rules, which are established on the Data Mining conclusions by logical connectives, gain a greater readability for human operators, who might draw their own conclusions by the attack situation and – additionally to FG's defense – block attack sources by other more simplistic means.
- The use of FG is computationally feasible and inexpensive: The implementation of the MySQL® server and the Java® runtime environment are free of charge and offers the advantages of the current system development paradigms.
- Manual intervention is possible: Whereas FG is an automatic system and reaches good results without any human intervention, this intervention on the other hand, is straightforward and may be executed at any time directly on the database.

## 5.2 Known Limitations

Some limitations of the employed algorithm's conclusion power may be studied in chapter 4. Though the results are – on the whole – very good, still the following may be improved:

- Fully correct treatment of legitimate connections under DDoS attacks has yet to be reached. Whereas an accuracy of 100% is the ultimate goal, methods should be found to lower the number of legitimate blocked connections as this implies in the production of DoS by the system.
- Dependency on parameters: FG yet depends on manually set parameters, which would have to be changed for further commercial versions in order to minimize the work of system administrators.
- The database has certain speed and space limitations. This is, however, true in any computational system. The data saved was maintained in a “compact” form (see appendix), thus not offering oversized demands of space and retrieval/storage speed. It should be said that the used data structure (Logical Fibering) does not present space limitations by its mathematical principles, which is one of its advantages.
- Results were obtained in simulation mode. This is due to the fact that real-life results are difficult to obtain and to test in non live environment. However, all effort has been undertaken to maintain results as plausible as possible. Because of the modular system structure, it may be expected that real-life results would look very similar to those obtained during simulation and that the main algorithms of the system were, in fact, validated as they do not presuppose a real-life environment.

## 5.3 Future Developments

In order to mature the implementation of FG the following steps should be taken in the future:

- Server tests should be executed. Mainly the jump in performance, which is expected from Personal Computer to servers, should be evaluated. Furthermore, the system must be exposed to such an attack situation as DoS attacks normally do not occur on machines of low economical interest (like home PCs) but on commercial sites with a series of high performance servers. In this case, especially the use inside clusters should be studied.
- Another thinkable development for the future might be an implementation in hardware/router firmware. This is especially true as routers use to be put at central entrance and exit points with several clients around them and an installation at such an instance would greatly reduce the need of customization and resulting vulnerabilities. Through an implementation in hardware, speed issues might be treated more easily, with the only possible tradeoff to have a smaller database (as this should be kept in memory and not on disc).
- The conclusion engine has to be fully rolled out (see Figure 5-1). Not only shall the Data Mining algorithm extract valid rules and mount them on the Secondary Data Structure. This Secondary Data Structure shall be further modeled to offer a more natural basis for human conclusions, e.g., indicating that a specific part of an IP-address and a minor part of an effect together most likely implied in an attack against the system.

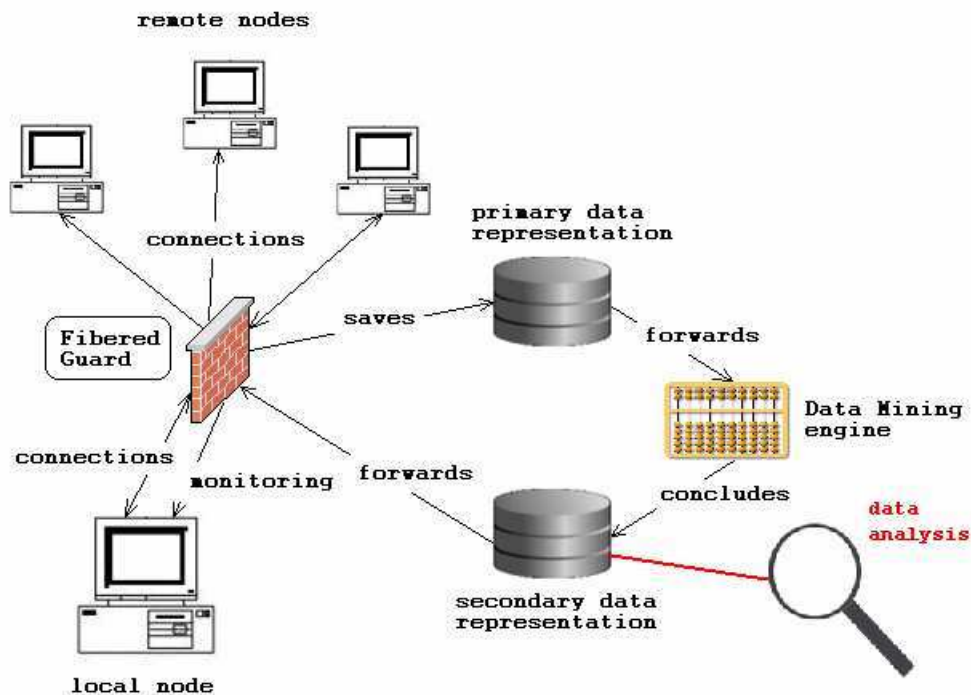


Figure 5-1: Data Analysis Tool

- Automatic Tuning: Currently a series of parameters exists, which was defined mainly on an empirical basis. A more systematic value calculation should be established and implemented in FG, so that the system always keeps itself at the best parameterization possible.
- Optimization has to be advanced. The necessities of performance should be attended through a greatly optimized way of data representation. This should be done by a different optimizing module substituting the current one. Out of the object oriented architecture, it is possible to maintain the rest of the implementation almost untouched.

- As stated in chapter 3, different Data Mining techniques or intelligent plugins may be used substituting the respective class in FG by the new implementation.
- Other database servers might be tested and used. The respective connection class can be easily substituted.
- Commercial application: A commercial application of FG's principles seems indicated as DoS and DDoS defense is a pressing security issue and up to now the field has not been satisfactorily covered. To achieve a commercial application it should be necessary to deploy the system firstly in a real life environment and discuss its merits in a long term analysis. As behavior similar to the simulation mode is expected, this might produce a powerful tool for the defense of commercial web sites.

## References

- [AMD64] G.M. Amdahl, G.A. Blaauw, F.P. Brooks, Jr.: Architecture of the IBM System/360, In: IBM Journal of Research and Development, Vol. 8, No. 2, 1964
- [APV05] A. Apvrille, M. Pourzandi, Secure Software Development by Example, IEEE Security & Privacy, vol. 3, no. 4, July/August, 2005, pp. 10-17
- [AUR00] T. Aura, P. Nikander, J. Leiwo: DOS-resistant authentication with client puzzles. In: B. Christianson, B. Crispo, M. Roe (eds.), Proceedings of the 8th International Workshop on Security Protocols, Cambridge/UK, 2000
- [BAK95] B.S. Baker, E. Grosse: Local Control over Filtered WWW Access, In: Fourth International World Wide Web Conference, Boston/USA, 1995
- [BAR64] P. Baran: On Distributed Communications. Memorandum RM-3420-PR <http://www.rand.org/publications/RM/RM3420/>, 1964
- [BAR99] Y. Bartal, A. Mayer, K. Nissim, A. Wool: Firmato: A Novell Firewall Management Toolkit, In: IEEE Symposium on Security and Privacy, pp. 17-31, 1999
- [BCR03] Bibliographical Center for Research: Remote Patron Authentication (RPA), <http://www.bcr.org/support/techsup/rpa.html>, 2003
- [BER00] M. Berry, G. Linoff: Mastering Data Mining, Hoboken/USA, 2000
- [BER97] M. Berry, G. Linoff: Data Mining Techniques for Marketing, Sales and Customer Support, Hoboken/USA, 1997
- [BIV02] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, M. Embrechts: Network Based Intrusion Detection using Neural Networks, Rensselaer Politechnic Institute, New York, 2002
- [BLA98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: An Architecture for Differentiated Services, Network Working Group, Request for Comments 2475, <http://asg.web.cmu.edu/rfc/rfc2475.html>, 1998
- [BMF02] Bundesministerium für Forschung und Bildung: Empfehlungen zum Schutz vor Verteilten Denial of Service Angriffen im Internet, <http://www.iid.de/netze/DoS.html>, 2002
- [BRA02] J. Branch, A. Bivens, C-Y. Chan, T-K. Lee and B.K. Szymanski: Denial of Service Intrusion Detection Using Time Dependent Deterministic Finite Automata. In: Proc. Graduate Research Conference, Troy, NY, 2002, pp. 45-51
- [BUR00] H. Burch, B. Cheswick: Tracing Anonymous Packets to Their Approximate Sources, In: 14<sup>th</sup> Systems Administration Conference (LISA2000), New Orleans/USA, 2000
- [CA05] Computer Associates International Inc.: Security Management – eTrust Web Access Control, <http://www3.ca.com/Solutions/Product.asp?ID=3224>, 2005



- [CAR00] D.R. Carvalho, A.A. Freitas: A hybrid decision tree/genetic algorithms for coping with the problem of small disjuncts in data mining, In: Processings of the Genetic and Evolutionary Computation Conference (GECCO2000), Las Vegas/USA, pp. 1061-1068, 2000
- [CER01] Cert Coordination Center: Denial of Service Attacks, [http://www.cert.org/tech\\_tips/denial\\_of\\_service.html](http://www.cert.org/tech_tips/denial_of_service.html), 2001
- [CER03] P.E. Ceruzzi: A History of Modern Computing, Cambridge/USA, 2003
- [CER07] Cert Coordination Center: Enterprise Security Management, <http://www.cert.org/esm/>, 2007
- [CHE05] W. R. Cheswick, S. M. Bellovin, A. D. Rubin: Firewalls e Segurança na Internet – Repelindo o hacker ardiloso, 2<sup>nd</sup> ed., Porto Alegre, 2005
- [CHU05] S. Cheung: Denial of Service against the Domain Name System: Threats and Countermeasures, CSL Technical Report SRI-CLS-05-02, Menlo Park/Canada, 2005
- [COA06] Sam Coates: Online casinos ‘used to launder cash’, In: TIMESONLINE, November 1, 2006
- [CRU04] F. da Cruz: The IBM 026 Key Punch, In: Columbia University Computing History, <http://www.columbia.edu/acis/history/punch.html>, 2004
- [CYB05] Cyberoam: Why Internet Access Management? <http://www.cyberoam.com/>, 2005
- [DEI04] H.M. Deitel, P.J. Deitel: Java – Como Programar, São Paulo, 2004
- [DEM90] A. Demers, S. Keshav, S. Shenker: Analysis and simulation of a fair queuing algorithm, In: Internetworking: Research and Experience, vol.1, no.1, pp.3--26, 1990
- [DFN01] DFN-CERT: Distributed Denial of Service Angriffe, <http://www.cert.dfn.de/infoserv/dib/dib-2000-01.html>, 2001
- [DIT99] D. Dittrich: The “stacheldraht” distributed denial of service attack tool, <http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt>, Washington/USA, 1999
- [DTT99] D. Dittrich: The “Tribe Flood Network” distributed denial of service attack tool, <http://staff.washington.edu/dittrich/misc/tfn.analysis>, Wasington/USA, 1999
- [DRE06] T. Dreier: Internetrecht, Skript zur Vorlesung an den Universitäten Freiburg und Karlsruhe, 2006
- [EKS03] G. Ekström: The Future of COBOL – An Acucorp View, [http://www.acucorp.com/company/press/acu\\_articles/The\\_Future\\_of\\_COBOL.pdf](http://www.acucorp.com/company/press/acu_articles/The_Future_of_COBOL.pdf), 2003
- [FAY96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth: From Data Mining to Knowledge Discovery: An Overview, In: Advances in Knowledge Discovery and Data Mining, pp. 1-34, 1996
- [FAY97] M. Fayyad, D.C. Schmidt: Object-Oriented Application Frameworks, In: Communications of the ACM, Special Issue on Object-Oriented Application Frameworks, Vol. 40, No. 10, October 1997

- [FER98] P. Ferguson, D. Senie: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, <http://rfc2267.x42.com/>, Natick/USA, 1998
- [FIS03] D. Fisher: Reactive to Offer Free Anti DoS Tool, <http://www.eweek.com/article2/0,1895,1660180,00.asp>, 2003
- [FLO93] S. Floyd, V. Jacobson: Random Early Detection gateways for Congestion Avoidance, V.1 N.4, 1993, p. 397-413
- [FLO99] S. Floyd, K. Fall, Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, 1999
- [FRÖ97] J. Fröhlich: Types of Neural Nets, <fbim.fh-regensburg.de/.../diplom/e-12-text.html>, 1997
- [GAI07] Global Analytik IT Services, [http://www.gaits.com/biometrics\\_face.asp](http://www.gaits.com/biometrics_face.asp), 2007
- [GAR06] L. Gardelli, M. Viroli, A. Omicini: On the Role of Simulations in Engineering Self-organizing MAS: The Case of an Intrusion Detection System in TuCSoN, In: S.A. Brueckner et al. (Eds.): ESOA 2001, pp. 153-166, Berlin, Heidelberg, 2006
- [GRA08] P. Grabosky: High Tech Crime: Information and Communication Related Crime: In: H.J. Schneider (ed.): International Handbook of Criminology. Volume 2. Berlin, New York : de Gruyter 2008 (forthcoming)
- [GUL00] Gulati, S.: The Internet Protocoll – Part II: The Present and the Future, <http://www.acm.org/crossroads/columns/connector/august2000.html>, 2000
- [GÜN62] G. Günther: Cybernetic Ontology and Transjunctional Operations. Biological Computer Lab. vol. 68 (Urbana III) In: Self-organizing Systems 1962, Washington/USA, 1962, pp. 313-392
- [HAY01] S. Haykin: Redes Neurais, Princípios e prática, 2nd ed., Porto Alegre, 2001
- [HOR04] U. Horstmann: Das Untier: Konturen einer Philosophie der Menschenflucht, Warendorf/Germany, 2004
- [HUA02] K.Y. Huang: Syntactic Pattern Recognition for Seismic Oil Exploration, Singapore, 2002
- [HUB04] M. Hubel: Technical Comparison of DB2 and MySQL, Hubel Consulting Inc., [ftp://ftp.software.ibm.com/software/data/pubs/papers/db2\\_mysql\\_comparison.pdf](ftp://ftp.software.ibm.com/software/data/pubs/papers/db2_mysql_comparison.pdf), 2004
- [HUS03] A. Hussain, J. Heidemann, C. Papadopoulos: A Framework for Classifying Denial of Service Attacks, ISI Technical Report 2003-569, 2003
- [IAA07] IAAIL: About the International Association for Artificial Intelligence and Law, <http://www.iaail.org/about/index.html>, 2007
- [IBM05] IBM Inc.: IBM Tivoli Access Manager for e-business, <http://www-306.ibm.com/software/tivoli/products/access-mgr-e-bus/>, 2005
- [IFR01] G. Ifrah: The Universal History of Computing: From the Abacus to the Quantum Computer, New York/USA, 2001

- [IOA00] S. Ioannidis, A.D. Keromytis, S.M. Bellovin, J.M. Smith: Implementing a Distributed Firewall, In: ACM Conference on Computer and Communications Security, pp. 190-199, 2000
- [JAN00] W. Jansen, P. Mell, T. Karygiannis, D. Marks: Mobile Agents in Intrusion Detection and Response. In: Proceedings of the 12<sup>th</sup> Annual Canadian Information Technology Security Symposium, Ottawa/Canada, 2000
- [JIN05] L. Jintao, N. Li, X. Wang, T. Yu: Denial of Service Attacks and Defenses in Decentralized Trust Management, CERIAS Tech Report 2005-76, Purdue University, West Lafayette, 2005
- [KAE98] S. D. Kaehler, Fuzzy Logic – An Introduction, <http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html>, 1998
- [KOH96] R. Kohavi, D. Sommerfield, J. Dougherty: Data mining using MLC++: A machine learning library of C++, In: Tools with Artificial Intelligence, IEEE Computer Press, pp. 234-245, 1996
- [KUM06] K. Kumar: An Integrated Approach for Defending Against Distributed Denial of Service (DDoS) Attacks, Indian Institute of Technology Roorkee, [http://www.cs.iitm.ernet.in/~iriss06/iitr\\_krishan\\_present.ppt](http://www.cs.iitm.ernet.in/~iriss06/iitr_krishan_present.ppt), 2006
- [LAU00] F. Lau, S. H. Rubin, M. H. Smith, L. Trajovic: Distributed denial of service attacks. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 2275-2280, Nashville/USA, 2000.
- [LAR04] D.T. Larose: Discovering Knowledge in Data: An Introduction to Data Mining, Hoboken/USA, 2004
- [LEE90] T. Berners-Lee, Information Management: A Proposal, <http://www.w3.org/History/1989/proposal.html>, 1990
- [LEE98] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium, 1998
- [LEI03] B.M. Leiner, V.G. Cerf, D.D. Clark, R.E. Kahn, L. Kleinrock, D.C. Lynch, J. Postel, L.G. Roberts, S. Wolff: A Brief History of the Internet, <http://www.isoc.org/internet/history/brief.shtml>, 2003
- [LEH03] S. Lehtonen, K. Ahola, T. Koskinen, M. Lyijynen, J. Pesola: Roaming Active Filtering Firewall. In: Processings of the European Smart Objects Conference (SOC'03), Grenoble, 2003
- [MAH01] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis and Vern Paxson, S. Shenker: Controlling high bandwidth aggregates in the network. Technical report, (draft), 2001
- [MAR04] R. Martin: Survey Indicates Identity and Access Management a Chief Security Concern, in: Entprise Networks & Servers, Austin/USA, 2004
- [MAR07] H.S. Markus: Internet Security Overview, <http://www.firewallguide.com/overview.htm>, 2007

[MCC03] S. McClure, J. Scambray, G. Kurtz: Hackers Expostos – Segredos e Solucoes para a Seguranca de Redes, 4th ed., Rio de Janeiro/Brasil, 2003

[MCK90] P. Mckenny: Stochastic Fairness Queuing. In: Proceedings of IEEE Infocom, IEEE Press, Piscataway, N.J., 1990, p. 733-740

[MET04] METAspectrum Market Summary: Data Mining Tools, [www.oracle.com/technology/products/bi/odm/pdf/odm\\_metaspectrum\\_1004.pdf](http://www.oracle.com/technology/products/bi/odm/pdf/odm_metaspectrum_1004.pdf), 2004

[MIR04] J. Mirkovic, P. Reiher: A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. ACM, 2004

[MIT95] M. Mitchell, S. Forrest: Genetic Algorithms and Artificial Life, In: C. Langton (ed.): Artificial Life: An Overview, Cambridge, London, 1995

[MIT02] J.C. Mitchell: Concepts in Programming Languages, Cambridge/UK, 2002

[MOR06] C.E. Morimoto: A Evolução dos Computadores, <http://www.guiadohardware.net/quias/15/printall.php>, 2006

[MYS03] MySQL AB: MySQL Performance Benchmarks: Measuring MySQL's Scalability and Throughput, A MySQL Technical White Paper, March 2005

[NIL96] N.J. Nilsson: Introduction to Machine Learning, Stanford/USA, 1996

[NSF06] NSFOCUS: NSFOCUS IDS/IPS & Collapsar Anti-DoS System, Editors' Choice Awarded by China Information World, <http://www.nsfocus.com/en/news/200611/281.html>, 2006

[OHA04] Ohama Public Library: Remote Patron Authentication, <http://www.omahapubliclibrary.org/rpa/webauth.exe?rs=netlib>, 2004

[OBL05] Oblix Inc.: Solutions for Web Access Management and Single Sign On – User Access Management for Secure Business Interactions, [http://www.oblix.com/solutions/access\\_management/](http://www.oblix.com/solutions/access_management/), 2005

[PAT98] D.A. Patterson, J.L. Hennessy: Computer Organization & Design: The Hardware / Software Interface, San Francisco, CA, USA, 1998

[PAZ92] M. Pazzani, D. Kibler: The utility of knowledge in inductive learning. In: Machine Learning, Vol. 9, pp. 57-94

[PER02] D. Peramunetilleke, K. Wong: Currency Exchange Rate Forecasting From News Headlines, In: Proceedings of the Thirteenth Australasian Database Conference (ADC2002), Melbourne/Australia, 2002

[PFA91] J.Pfalzgraf: Logical fiberings and polycontextural systems. In: Ph.Jorrand and J.Kelemen (eds.): Fundamentals of Artificial Intelligence Research FAIR'91. Proceedings, Springer LNCS 535, Subseries Lecture Notes in AI, 1991

[PFA95] J.Pfalzgraf, V.Sofronie: Decomposing Many-valued Logics: An Experimental Case Study, RISC-Linz Report Series 95-44, J. Kepler University, Linz/Austria, 1995

- [PFA01] Pfalzgraf, J.: The concept of Logical Fiberings and Fibered Logical Controllers. In: Proceedings Computing Anticipatory Systems: CASYS 2000. Liège, Belgium. AIP Conference Proceedings, Vol.573, 2001, pp. 683-693
- [PFA04] Pfalzgraf, J., Edtmayr, J.: The Concept of Logical Fiberings: Distributed Logics for Multiagent Systems. In: Proceedings 17th European Meeting on Cybernetics and Systems Research (EMCSR'2004) Vienna, 2004
- [PFA00] Pfalzgraf, J., Meixl, W.: A Logical Approach to Model Concurrency in Multiagent Systems. In: Proceedings 15th European Meeting on Cybernetics and Systems Research (EMCSR'2000), Vienna, 2000
- [PFA96] Pfalzgraf, J., Sigmund, U., Stokkermans, K.: Towards a general approach for modeling actions and change in cooperating agents scenarios. In: Special Issue of IGPL (Journal of the Interest Group in Pure and Applied Logics), IGPL 4 (3),1996, pp. 445-472
- [RAY07] M.D. Ryan: Firewalls and tunnels, <http://www.cs.bham.ac.uk/~mdr/teaching/modules/netsec/lectures/firewalls+tunnels.html>, 2007
- [REZ03] Rezende, S. O. (Coord.): Sistemas Inteligentes – Fundamentos e Aplicações, Barueri-SP, Brazil, 2003
- [ROB06] M.M. Roberts: Lessons for the Future Internet – Learning from the Past, In: EDUCAUSEreview, July/August, 2006
- [ROE03] P. J. Roebber, S. L. Bruening, D. M. Schultz, J. V. Cortinas Jr.: Improving snowfall forecasting by diagnosing snow density. In: Wea. Forecasting, Vol. 18, 2003, pp. 264-287.
- [ROL99] Rowland, Todd: Fiber Bundle. From: MathWorld - A Wolfram Web Resource, created by Eric W. Weisstein. <http://mathworld.wolfram.com/FiberBundle.html>, 1999
- [ROW99] Rowland, Todd: Fiber Bundle. From: MathWorld - A Wolfram Web Resource, created by Eric W. Weisstein. <http://mathworld.wolfram.com/VectorBundle.html>, 1999
- [RSA05] RSA Security Inc.: Web Access Management, <http://www.rsasecurity.com/node.asp?id=1185>, 2005
- [SAB95] R.M.E. Sabbatini: A história do Windows, In: Jornal Correio Popular, Caderno Informática, 17/10/95, Campinas-SP, Brazil
- [SAM72] J.E. Sammet: Programming Languages: History and Future, In: Communications of the ACM, Vol. 15, Nr. 7, July 1972, pp. 601-610
- [SAN06] J.A. Sánchez: Introduction to Pattern Recognition, Universidad Politécnica de Valencia/Espanha, <http://www.dsic.upv.es/~jandreu>, 2006
- [SAN00] SANS Institute: Global Incident Analysis Center – Special Notice – Egress Filing v 0.2, <http://www.sans.org/y2k/egress.htm>, Maryland/USA, 2000
- [SAV00] S. Savage, D. Wetherall, A. Karlin, T. Anderson. Practical network support for ip traceback. In: Proceedings of the 2000 ACM SIGCOMM Conference, Uppsala/Sweden, 2000

[SCA07] B. Scalzo: The Top Five Database Benchmarking Questions, Quest Software White Paper, Aliso Viejo/USA, 2007

[SCC05] M.O. Schneider, J. Calmet: Fibered Guard – A Hybrid Intelligent Approach to Denial of Service Prevention, In: Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05), Vienna/Austria, 2005

[SCH05] J. Schmidt. Dämme gegen die SYN-Flut, <http://www.heise.de/security/artikel/43066>, 2005

[SCH87] H.J. Schneider: Kriminologie, Berlin/New York, 1987

[SCR03] M.O. Schneider, J.L.G. Rosa: Neural Labyrinth Robot - Finding the Best Way in a Connectionist Fashion, IV Encontro Nacional de Inteligência Artificial (ENIA'03), Campinas, Brazil, 2003, pp. 1751-1760

[SEC07] SecureSynergy Consulting: Consulting Practice, <http://www.securesynergy.com/consulting/securityneedsplanning.php>, 2007

[SEC03] Secure Synergy Consulting: Autonomic Systems – Combating DDoS Attacks, <http://www.securesynergy.com/library/articles/037-2003.php>, 2003

[SEI04] J.W. Seiffert, CSR Report for Congress, Oder Code RL31798, Data Mining: An Overview, The Library of Congress, 2004

[SHA97] V. Shah, M. Sivitanides, R. Martin: Pitfalls of Object-Oriented Development, <http://www.westga.edu/~bquest/1997/object.html>, 1997

[SHI02] C. Shields: What do we mean by Network Denial of Service? In: Proceedings of the 2002 IEEE Workshop on Information Assurance and Security, West Point/USA, 2002

[SON07] SonicWALL Inc.: SonicWALL – Comprehensive Internet Security, <http://www.sonicwall.com/us/>, 2007

[SPI07] C. Spieth: Mathematical Models – Chapter 3: Algorithms, <http://www-ra.informatik.uni-tuebingen.de/software/JCell/tutorial/ch03s03.html>, 2007

[STE51] N. Steenrod: The Topology of Fibre Bundles, Princeton/USA, 1951

[STO98] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks, June 1998. Technical Report CMU-CS-98-136

[STO00] R. Stone: CenterTrack: An IP Overlay Network for Tracking DoS Floods, In: Proceedings of the 9th USENIX Security Symposium, Denver/USA, 2000

[STU04] W. Sturgeon: Fear of viruses and poor protection grows, In: CNET News.com, July 6, 2004

[SUP05] SupportLINK Systems Inc.: Firewalls, <http://www.supportlink.ca/92.html>, 2005

- [TOH06] J. Tohka: Introduction to Pattern Recognition, Technical Report SGN-2506, Tampere University of Technology, Finland, 2006
- [TUP03] U. K. Tupakula, V. Varadharajan: A practical method to counteract denial of service attacks, In: proceedings of the twenty-fifth Australasian computer science conference (ACSC2003), Adelaide/Australia, 2003
- [ULB03] H. C. Ulbrich, J. Della Valle: Universidade Hacker – Desvende todos os segredos do submundo dos hackers, 2<sup>nd</sup> ed., Sao Paulo/Brasil, 2003
- [VW07] Viking Waters Inc.: A Short History of Microcomputers, <http://www.vikingwaters.com/htmlpages/PCHist.htm>, 2007
- [W3C01] W3C®: About the World Wide Web, <http://www.w3.org/WWW/>, 2001
- [WAC02] J. Wack, K. Cutler, J. Pole: Guidelines on Firewalls and Firewall Policy, Recommendations of the National Institute of Standards and Technology (NIST), Gaithersburg/USA, 2002
- [WAL06] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, S. Shenker: DDoS Defense by Offense, In: Proceedings of the SIGCOMM'06, Pisa/Italy, 2006
- [WEY07] S. Weyhrich: Apple II History, <http://apple2history.org/>, 2007
- [WEI03] E. W. Weisstein: Vector Space. From: MathWorld - A Wolfram Web Resource. <http://mathworld.wolfram.com/VectorSpace.html>, 2003
- [WIL01] P. Williams: VT100 User Guide, <http://vt100.net/docs/vt100-ug/>, 2001
- [WIL02] P. Williams: Organized Crime and Cybercrime: Synergies, Trends, and Responses, <http://usinfo.state.gov/journals/itgic/0801/ijge/gj07.htm>, 2002
- [WIL04] E.v.Wiltenburg: Routers & Firewalls, lecture at the University of Victoria, British Columbia/Canada 2004
- [WIT00] I. H. Witten, E. Frank: Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations, San Francisco, San Diego, New York, Boston, London, Sydney, Tokyo, 2000
- [WIT06] A. Witzel: Neural-Symbolic Integration - Constructive Approaches, Technical Report WV-06-03, Knowledge Representation and Reasoning Group, Artificial Intelligence Institute, Department of Computer Science, TU Dresden, 2006
- [WNE94] J. Wnek: Comparing Symbolic and Sub-Symbolic Learning, In: Machine Learning: A Multistrategy Approach, Vol. IV, San Francisco/USA, 1994
- [XIO01] Y. Xiong, S. Liu, P. Sun: On the defense of the Distributed Denial of Service Attacks: An On-Off Feedback Control Approach, IEEE Transactions on System, Man and Cybernetics-Part A: Systems and Humans. Vol. 31 No.4, July 2001

[XU04] Y. Xu, H.C.J. Lee: A Source Address Filtering Firewall to Defend against Denial of Service Attacks, In: Proceedings of the Vehicular Technology Conference (VTC2004-Fall), Vol. 5, pp. 3296-3300, 2004

[YUR07] C.E. Yurkanan: A Technical History of the ARPANET – A Technical Tour, [http://www.cs.utexas.edu/users/chris/think/ARPANET/Technical\\_Tour/overview.shtml](http://www.cs.utexas.edu/users/chris/think/ARPANET/Technical_Tour/overview.shtml), 2007

[ZAI99] O.R. Zaïane: CMPUT690 Principles of Knowledge Discovery in Databases, University of Alberta, Canada, 1999

[ZEL01] J. Zeleznikow, A. Stranieri: Technical and Legal Issues in E-Commerce: An Intelligent Systems Perspective, <http://www.cs.wustl.edu/icail2001/tutoriala2.html>, 2001

[ZUS03] H. Zuse: Konrad Zuse und seine Rechner, [http://irb.cs.tu-berlin.de/~zuse/Konrad\\_Zuse/index.html](http://irb.cs.tu-berlin.de/~zuse/Konrad_Zuse/index.html), 2003

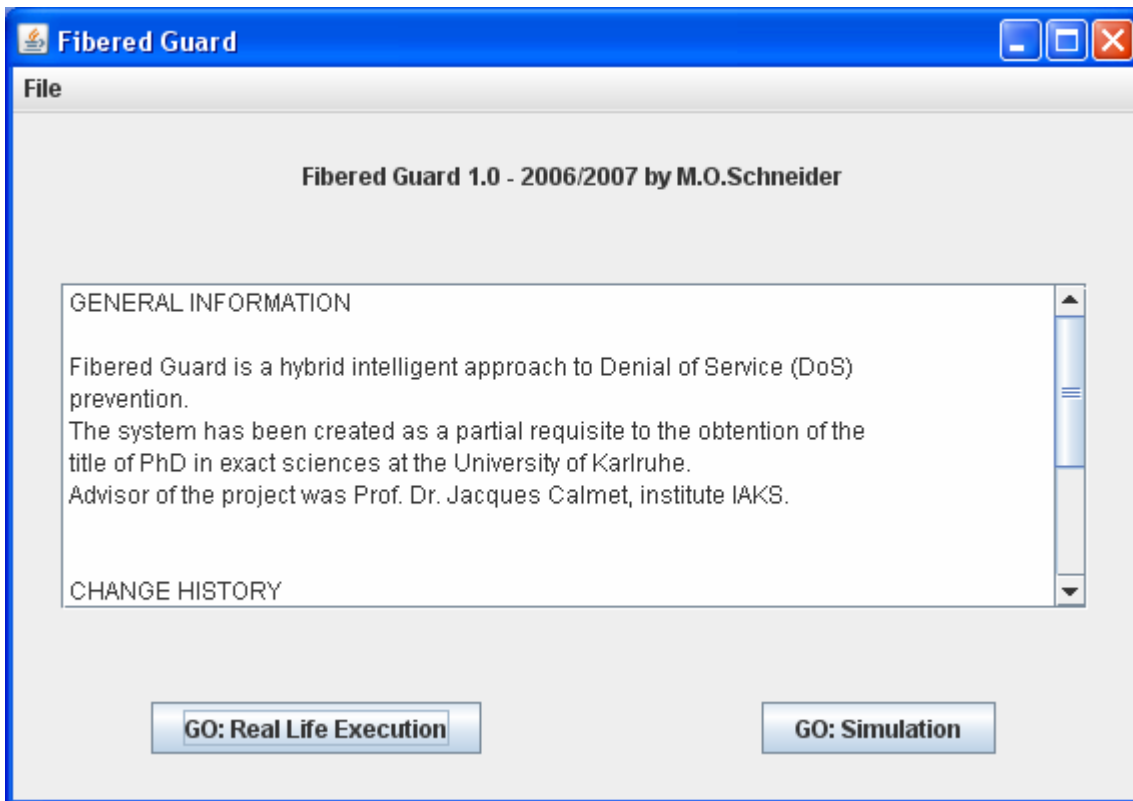


# Appendix

## Appendix A: Interface Description

### Main Screen

After installation and the first run of the system, the following screen should be shown:



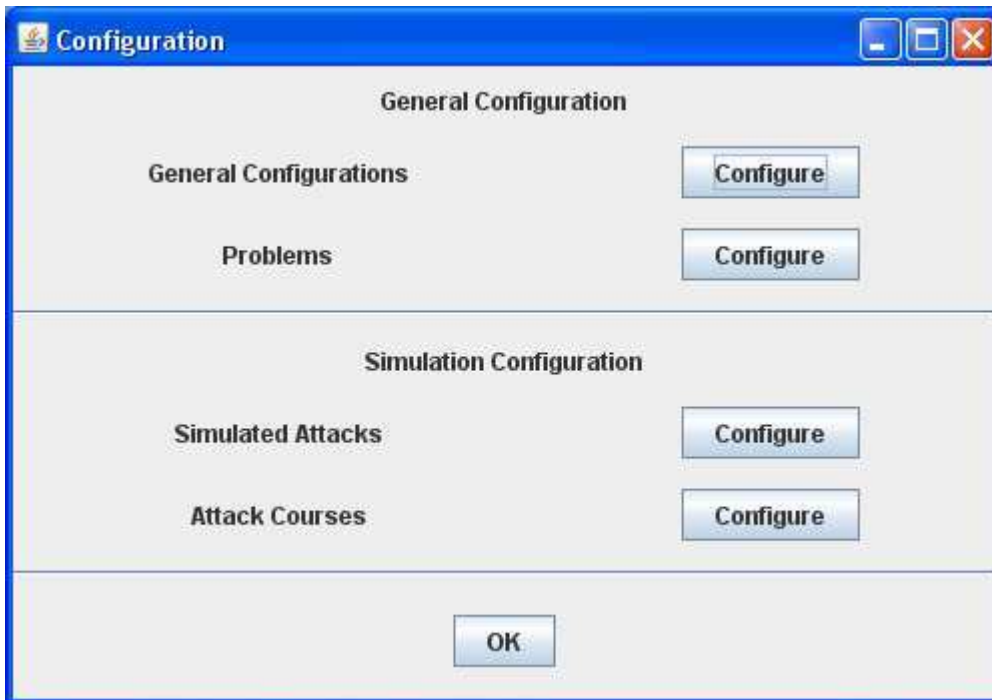
Apart from the general information given in the splash window, which might be read, it is also possible to reach the following functions:

File (upper left corner): In the drop-down menu, which is opened, the user may chose to change the general configuration of the system.

GO: Real Life Execution: Start of real-life defense with Fibered Guard (i.e., against real DoS and DDoS attacks). This mode is silent, meaning that after its start the current window is closed. If necessary, it may be reached again typing in the word "exit" on the console window. Attention: Closing the console window closes the application.

GO: Simulation: Start of Fibered Guard's test environment.

## Configurations



If the user chooses to enter the configuration mode on the main screen, the above screen is shown with the following options:

Clicking "Configure" on the right side of "General Configurations" leads to the general configurations window, where specific system operation parameters can be configured.

Clicking "Configure" on the right side of "Problems" leads to a window where problem conditions of system resources may be defined (thus setting the decision of "normal", "slight problem mode" and "severe problem" mode of the system).

Clicking "Configure" on the right side of "Simulated Attacks" leads to the definition of single attacks.

Clicking "Configure" on the right side of "Attack Courses" leads to the window, where attack courses may be defined.

**General Configurations**

**Delays**

Response Delay: 0

Iteration Delay: 0

**Execution Module**

False Ping Probability: 0

No Ping Block Ratio: 0

Normal Operation Saving Ratio: 0

Severe Problem Block Ratio: 0

**ANN**

Hidden Layer Neurons: 0

Learning Rate: 0

**DM Plugin**

Min Rule Precision: 0

Min Elements Per Rule: 0

Nr Rules Apply: 0

**LF Connectives**

LF Builder: on

**DB Configuration**

Connection Data Validity: 0

Rule Data Validity: 0

OK Cancel

In the above window for general configurations a series of system parameters may be set. A description of several of these parameters is given in the section “system parameters” in this appendix.

**Problem Definition**

ID: 1

Name: CPU Clock

Description: Current use of CPU cycles

Unit: Percentage

Slight Problem: 70

Severe Problem: 85

Available: 100

Jump: 5

< > New Insert Change Delete Cancel

In the problem definition window, problem conditions may be defined for a set of system resources. All monitored system resources must be parameterized as the definition of the system mode (as explained in chapter 3) depends on this definition.

**Attacks**

ID: 5

Name: DDoS2

Description: Using UDP protocol, other different effects:

Protocol: UDP

Source IP:   Random

Answer to Ping:

Destination IP:   Random

Source DNS:   Random

Destination DNS:   Random

Program:   Random

Direction: Incoming

1. Effect: Disk Load

1. Impact: 0  Random

2. Effect: RAM Usage

2. Impact: 0  Random

3. Effect: CPU Clock

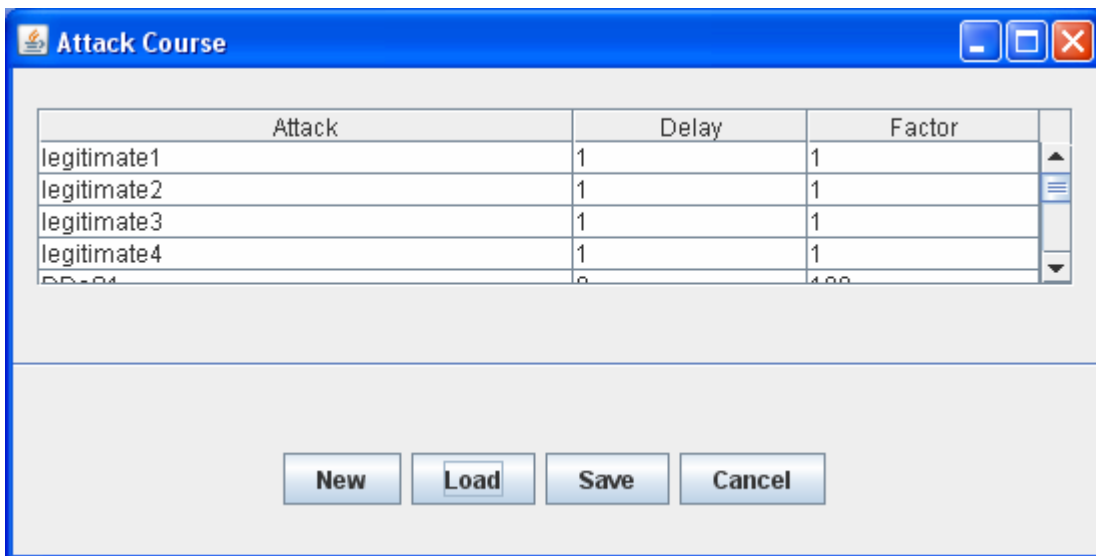
3. Impact: 0  Random

Iterations: 5  Random

< > New Insert Change Delete Cancel

The attack definition window offers several facilities to set simulated attacks. Name and description only serve for information purposes and are not considered by the simulation's defense routine. Several attack characteristics (IP and DNS information, program etc.) may be given and the effect the attack has on simulated system resources can be defined giving up to 3 different effects. At every entry with a "Random" checkbox

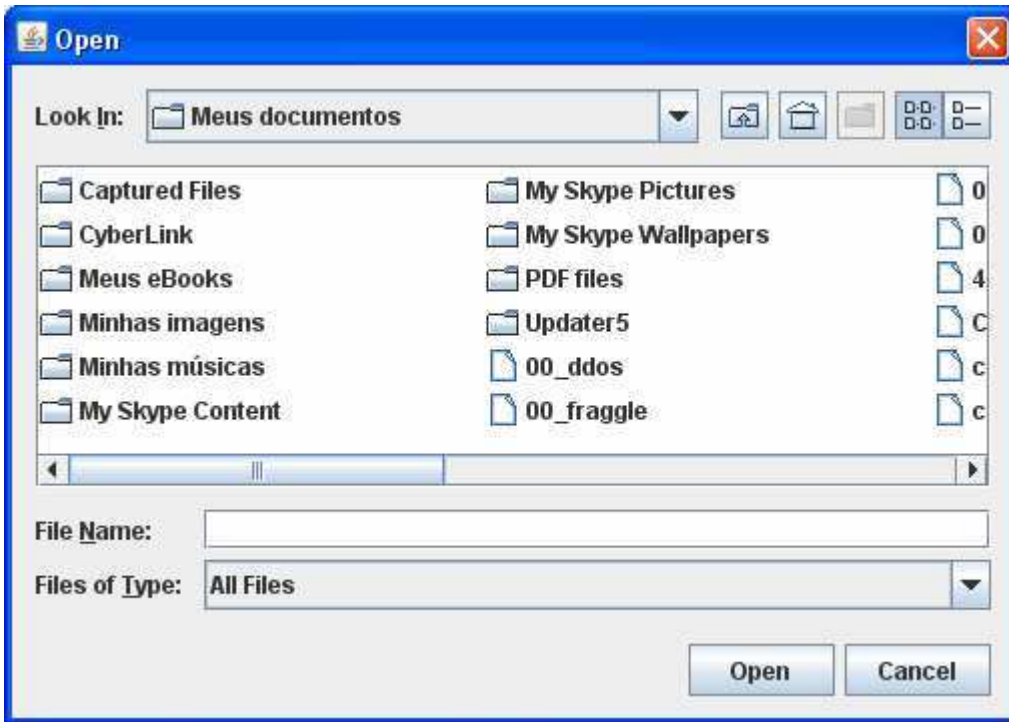
at the side, the data may be randomized (i.e., will be different with every new simulated connection of the specific type).



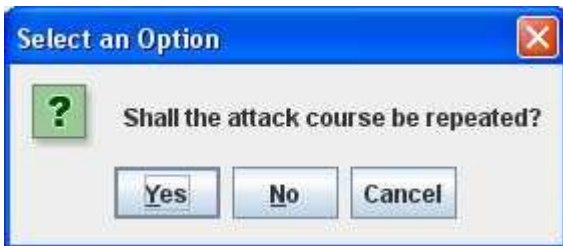
Finally, an attack course may be defined, i.e. a sequence of simulated attacks, giving delay in iterations (whereas 0=no delay) between two attacks and factor (i.e., how many times the specific attack is applied). It should be noted that legitimate connections can be mixed into the attack course in order to evaluate their blocking.

### Attack Simulation

Pressing “GO: Simulation” on the main window, firstly an attack course has to be provided. Therefore a file selector is opened and the user may load a pre-defined attack course.



After the definition of the attack course, two specific runtime configurations have to be made:



An attack course can be repeated or can simply run to its end. If a repetition is desired, the user should answer the respective question with “yes”.



Normally Fibered Guard will defend against the simulated attacks. If, however, the objective is to test the genuine impact of an attack without a defense mechanism, the user may answer the respective question with “no”.

Finally the window for attack simulation is opened:

The screenshot shows a window titled "Simulated Execution" with a blue title bar. The window contains several input fields and buttons. At the top, there is an "Iteration" field with the value "1". To its right are two green labels: "repeat" and "defense". Below these is a table with three columns: "Available", "Total", and "Used (%)". The table has seven rows of resource data, each with a green label and three input fields. The resources listed are CPU Clock, RAM Usage, Network Bandwidth, Disk Load, Open Connections, and Page File Usage. Each resource has an "Available" value of 100, a "Total" value of 100, and a "Used (%)" value of 0. Below the table are four rows of "(void)" labels, each followed by an empty input field. At the bottom of the window, there is a "Mode" field with the value "Normal Operation" and a "Delay" field with the value "100". At the very bottom, there are five buttons: "Go!", "Trace", "Stop", "Reset", and "Cancel".

	Available	Total	Used (%)
CPU Clock	100	100	0
RAM Usage	100	100	0
Network Bandwidth	100	100	0
Disk Load	100	100	0
Open Connections	100	100	0
Page File Usage	100	100	0
(void)			
(void)			
(void)			
(void)			

It presents all system resources, which are monitored, giving how many units are currently available, which is the total (maximum) value of available units and the respective percentage of use.

Furthermore the current system mode is shown (normal, slight problem, severe problem, upon dos).



The user may start the simulation by pressing “go!”, trace into the simulation (i.e., go one iteration forward) clicking “trace”, stop a running simulation with “stop”, reset the virtual machine to its initial values with “reset” or cancel the simulation with “cancel”.

During simulation several internal data for debugging are shown on console and a pre-defined log-file is written.

## Appendix B: Database Configuration

Fibered Guard makes use of the following productive tables:

Table: **simulated\_attacks**

Description: Related to the connections (mostly attacks) invoked during the simulation mode.

id int unsigned not null auto_increment,	:	internal ID
name varchar(80) not null,	:	connection short name
description varchar(255),	:	description
protocol varchar(40),	:	protocol used
source_ip varchar(80),	:	source IP data
ping varchar(80),	:	response to ping (y/n)
destination_ip varchar(80),	:	destination IP
source_dns varchar(80),	:	source DNS
destination_dns varchar(80),	:	destination DNS
program varchar(80),	:	program
direction varchar(40) not null,	:	incoming/outgoing
effect1 int unsigned not null,	:	first effect
impact1 int not null,	:	impact of first effect
effect2 int unsigned,	:	second effect
impact2 int,	:	impact of second effect
effect3 int unsigned,	:	third effect
impact3 int,	:	impact of third effect
primary key (id),	:	table primary key
foreign key (effect1) references system_characteristics (id),	:	effects
foreign key (effect2) references system_characteristics (id),	:	correspond
foreign key (effect3) references system_characteristics (id),	:	to system
		: characteristics
iterations int;	:	iterations connection stays in

Table: **general\_configurations**

Description: Used for all system parameter values.

id int unsigned not null auto_increment,	:	internal id
name varchar(80) not null,	:	name of the parameter
value int not null,	:	value of the parameter
primary key(id)	:	table primary key

Table: **system\_characteristics**

Description: Monitored system resources.

id int unsigned not null auto_increment,	:	internal id
name varchar(80) not null,	:	name of resource
description varchar(255),	:	description of resource
unit varchar(80) not null,	:	unit (discrete, percent)
slight_problem int not null,	:	slight problem limit
severe_problem int not null,	:	severe problem limit
available int,	:	total available
jump int,	:	jump in usage
primary key (id)	:	table primary key

Table: **signature\_fibering**

Description: Primary Data Representation.

id int unsigned not null auto_increment,	:	internal id
heading varchar(500),	:	fiber codification (1)
data varchar(1000),	:	fiber codification (2)
situation varchar(20),	:	situation on connect
creation timestamp not null,	:	used for validity
primary key (id)	:	table primary key

Table: **conclusion\_fibering**

Description: Secondary Data Representation

id int unsigned not null auto_increment,	:	internal id
heading varchar(500),	:	fiber codification (1)
data varchar(1000),	:	fiber codification (2)
situation varchar(20),	:	situation on connect

creation timestamp not null, : used for validity  
 primary key (id) : table primary key

## Appendix C: Parameters

The following general parameters are used in Fibered Guard (name, default value and description):

NO\_PING\_BLOCK\_RATIO = 25 : Connection blocked on "no ping" (%)  
 NORMAL\_OPERATIONS\_SAVING\_RATIO = 80 : Data saving (%) on normal operation  
 SEVERE\_PROBLEM\_SAVING\_RATIO = 90 : Data saving (%) on severe problem  
 SEVERE\_PROBLEM\_BLOCK\_RATIO = 90 : Random Blocking (%) on severe prob.  
 PLUGIN\_ALGORITHM = DM\_ALG : Definition of plugin algorithm  
 HIDDEN\_LAYER\_NEURONS = 15 : Neurons in the ANN hidden layer  
 LEARNING\_RATE = 0.35 : ANN Learning rate  
 BACKPROPAGATION\_REPETITION = 10 : ANN Learning repetitions  
 MIN\_RULE\_PRECISION = 75 : Minimal rule precision (DM) (%)  
 MIN\_ELEMENTS\_PER\_RULE = 1000 : Minimum elements per rule (DM)  
 ANALYSIS\_RATIO = 70 : Analysis ratio (DM) (%)  
 UPDATE\_RULES\_RATIO = 50 : Update rules ratio (DM) (%)  
 CONNECTION\_DATA\_VALIDITY = 10000 : Validity of connection data (DB)  
 RULE\_DATA\_VALIDITY = 10000 : Validity of rule data (DB)

## Appendix D: Class and method details

- **Class ANNPlugin**

Implementation of the Artificial Neural Network, using a Multiplayer Perceptron with standard Backpropagation for learning.

Constructor Summary	
<a href="#">ANNPlugin</a>	( <a href="#">Database</a> db, int hiddenLayer, double learningRate)
	Constructor of the ANN class
Method Summary	
void	<a href="#">backPropagation</a> ()
	Implementation of the Backpropagation algorithm for learning

void	<a href="#"><u>compareToDB()</u></a> Tests the ANN function towards the database with the objective of assessing its performance
void	<a href="#"><u>currentAttackToInputs</u></a> ( java.lang.String head, java.lang.String data) Converts the current attack read from the database into the binary input layer
void	<a href="#"><u>dumpAll</u></a> () Dumps all values of the ANN (por debug)
void	<a href="#"><u>dumpInOut</u></a> () Dumps input values and output activations to console (for debug)
void	<a href="#"><u>dumpInputs</u></a> () Dumps the input layer values to console (for debug purposes)
void	<a href="#"><u>dumpOutputs</u></a> () Dumps the output layer activations to console (for debug purposes)
void	<a href="#"><u>flushAll</u></a> () Sets all activation values to zero and generates random values between 0.1 and -0.1 for all synapses
void	<a href="#"><u>flushDesiredOutput</u></a> () Sets the desired outputs (training phase) to zero
void	<a href="#"><u>flushHiddenOutput</u></a> () Sets hidden and output neuron activation values to zero
int	<a href="#"><u>getSuggestedOutput</u></a> () Gets the output, with the greatest value, i.e., the conclusion suggested by the ANN in the current situation
void	<a href="#"><u>learnFromDB</u></a> () Picks up data sets from the database and learns them
static void	<a href="#"><u>main</u></a> ( java.lang.String[] args) Side entrance for the purpose of isolated testing
void	<a href="#"><u>propagate</u></a> () Propagation of the ANN; activation is calculated using the sigmoid function
double	<a href="#"><u>sigmoid</u></a> (double value) Calculates the sigmoid for a value.

- **Class AttackCourseDefinition**

This class administrates the definition of attack courses

Nested Class Summary	
(package private) class	<a href="#"><u>AttackCourseDefinition.CancelListener</u></a> Inner class, which closes the current window
(package private) class	<a href="#"><u>AttackCourseDefinition.CourseTableModel</u></a> Inner class, which is used for the creation of the table inside the window
(package private) class	<a href="#"><u>AttackCourseDefinition.LoadListener</u></a> Inner class, which loads an existing attack course from disc
(package private) class	<a href="#"><u>AttackCourseDefinition.NewListener</u></a> Inner class for the definition of a new attack course
(package private) class	<a href="#"><u>AttackCourseDefinition.SaveListener</u></a> Inner class, which is responsible for saving the attack course defined by the user to disc

Constructor Summary	
<a href="#">AttackCourseDefinition</a> ( <a href="#">Database</a> databaseObj, <a href="#">Log</a> logObj)	
Constructor, in which the window is created	
Method Summary	
java.sql.ResultSet	<a href="#">getAttacks</a> () Gets the names of all simulated attacks from the database

- **Class AttackCourseDefinition.CancelListener**

Constructor Summary	
<a href="#">AttackCourseDefinition.CancelListener</a> ()	
Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class AttackCourseDefinition.CourseTableModel**

Inner class, which is used for the creation of the table inside the window

Fields inherited from class javax.swing.table.AbstractTableModel
listenerList

Constructor Summary	
<a href="#">AttackCourseDefinition.CourseTableModel</a> ()	
Method Summary	
int	<a href="#">getColumnCount</a> ()
java.lang.String	<a href="#">getColumnName</a> (int col)
int	<a href="#">getRowCount</a> ()
java.lang.Object	<a href="#">getValueAt</a> (int row, int col)
void	<a href="#">init</a> ()
boolean	<a href="#">isCellEditable</a> (int row, int col)
void	<a href="#">setValueAt</a> (java.lang.Object value, int row, int col)

#### Methods inherited from class javax.swing.table.AbstractTableModel

addTableModelListener, findColumn, fireTableCellUpdated, fireTableChanged, fireTableDataChanged, fireTableRowsDeleted, fireTableRowsInserted, fireTableRowsUpdated, fireTableStructureChanged, getColumnClass, getListeners, getTableModelListeners, removeTableModelListener

- **Class AttackCourseDefinition.LoadListener**

Inner class, which loads an existing attack course from disc

#### Constructor Summary

[AttackCourseDefinition.LoadListener\(\)](#)

#### Method Summary

void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)
------	---

java.lang.String	<a href="#">getFirstArgument</a> (java.lang.String s1)
------------------	--

java.lang.String	<a href="#">getSecondArgument</a> (java.lang.String s1)
------------------	---

java.lang.String	<a href="#">getThirdArgument</a> (java.lang.String s1)
------------------	--

- **Class AttackCourseDefinition.NewListener**

#### Constructor Summary

[AttackCourseDefinition.NewListener\(\)](#)

#### Method Summary

void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)
------	---

- **Class AttackCourseDefinition.SaveListener**

Inner class, which is responsible for saving the attack course defined by the user to disc

#### Constructor Summary

[AttackCourseDefinition.SaveListener\(\)](#)

#### Method Summary

void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)
------	---

void	<a href="#">saveFile</a> ()
------	-----------------------------

- **Class Configuration**

Nested Class Summary	
(package private) class	<a href="#">Configuration.CourseListener</a> Inner class, which creates in instance of AttackCourseDefinition
(package private) class	<a href="#">Configuration.GeneralListener</a> Inner class, which instantiates the GeneralConfig
(package private) class	<a href="#">Configuration.okListener</a> Inner class, which hides the current window
(package private) class	<a href="#">Configuration.ProblemsListener</a> Inner class, which instantiates SystemCharacteristicDefinitions
(package private) class	<a href="#">Configuration.SimulatedAttacksListener</a> Inner class, which creates in instance of SimulatedAttacksDefinitions

Constructor Summary	
	<a href="#">Configuration</a> ( <a href="#">Database</a> databaseObj, <a href="#">Log</a> logObj) Constructor with the definition of the window and buttons
Method Summary	
void	<a href="#">addActionListeners</a> () Adds the ActionListener to the buttons
void	<a href="#">arrangeWindow</a> () Mounts the items inside the window

- **Class Configuration.CourseListener**

Inner class, which creates in instance of AttackCourseDefinition

Constructor Summary	
	<a href="#">Configuration.CourseListener</a> ()

Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class Configuration.GeneralListener**

Inner class, which instantiates the GeneralConfig

Constructor Summary	
	<a href="#">Configuration.GeneralListener</a> ()

<b>Method Summary</b>	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class Configuration.okListener**

Inner class, which hides the current window

<b>Constructor Summary</b>	
<a href="#">Configuration.okListener</a> ()	
<b>Method Summary</b>	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class Configuration.ProblemsListener**

Inner class, which instantiates SystemCharacteristicDefinitions

<b>Constructor Summary</b>	
<a href="#">Configuration.ProblemsListener</a> ()	
<b>Method Summary</b>	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class Configuration.SimulatedAttacksListener**

Inner class, which creates an instance of SimulatedAttacksDefinitions

<b>Constructor Summary</b>	
<a href="#">Configuration.SimulatedAttacksListener</a> ()	
<b>Method Summary</b>	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class ConnectionItem**

Class, which defines an item of connections, mainly consisting of getters and setters

<b>Constructor Summary</b>	
<a href="#">ConnectionItem</a> ()	
<b>Method Summary</b>	
	void <a href="#">convertToNumbers</a> ()



	Conversion of factor and delay from string to integer
void	<a href="#">convertTotalStr()</a> Separation of the total string in attack, delay and factor
int	<a href="#">getAttackNr()</a>
java.lang.String	<a href="#">getAttackStr()</a>
int	<a href="#">getDelayNr()</a>
java.lang.String	<a href="#">getDelayStr()</a>
int	<a href="#">getFactorNr()</a>
java.lang.String	<a href="#">getFactorStr()</a>
java.lang.String	<a href="#">getTotalStr()</a>
void	<a href="#">setAttackNr(int value)</a>
void	<a href="#">setAttackStr(java.lang.String value)</a>
void	<a href="#">setDelayNr(int value)</a>
void	<a href="#">setDelayStr(java.lang.String value)</a>
void	<a href="#">setFactorNr(int value)</a>
void	<a href="#">setFactorStr(java.lang.String value)</a>
void	<a href="#">setTotalStr(java.lang.String value)</a>

- **Class ConnectionPool**

Class, which treats current connections to the virtual machine in the form of a pool

Constructor Summary	
<a href="#">ConnectionPool()</a>	Definition of maximum items in pool and initialization
Method Summary	
void	<a href="#">auxToOrder()</a> Auxiliary array converted to order array
void	<a href="#">dump()</a> Prints the connection pool items on console (for debug)
<a href="#">ConnectionPoolItem[]</a>	<a href="#">flush(ConnectionPoolItem[] aux2)</a> Resets all items in the connection pool
<a href="#">ConnectionPoolItem</a>	<a href="#">getNext()</a> gets the next connection to be released
void	<a href="#">orderToAux()</a>

	Order array converted to auxiliary array
void	<a href="#">push</a> (java.lang.String name, int ID, int impact, int delay) Adds a connection to the pool
void	<a href="#">reorg</a> () Reorganization of arrays: deletion of released orders
void	<a href="#">restore</a> () sets the release index to zero
void	<a href="#">tick</a> () one step forward in time (diminish all delays)

- **Class ConnectionPoolItem**

Data type for an item in the connection pool (i.e. a connection to the virtual machine). Used in simulation mode only.

Constructor Summary	
	<a href="#">ConnectionPoolItem</a> ()
Method Summary	
int	<a href="#">getDelay</a> ()
int	<a href="#">getEffectID</a> ()
java.lang.String	<a href="#">getEffectName</a> ()
int	<a href="#">getImpact</a> ()
void	<a href="#">setDelay</a> (int delay)
void	<a href="#">setEffectID</a> (int effectID)
void	<a href="#">setEffectName</a> (java.lang.String effectName)
void	<a href="#">setImpact</a> (int impact)

- **Class Database**

Class, which treats the connection with the database on a high (not mySQL specific) level

Constructor Summary	
	<a href="#">Database</a> () Constructor with the creation of the connection as well as three statements (to be used in parallel)
Method Summary	
void	<a href="#">error</a> (java.lang.String inputStr) Shows a database error in a message box
java.sql.ResultSet	<a href="#">executeQuery</a> (java.lang.String inputStr) Executes a query on the first statement
java.sql.ResultSet	<a href="#">executeQueryAux</a> (java.lang.String inputStr) Executes a query on statement3

void	<a href="#">executeUpdate</a> ( java.lang.String inputStr) Routine, which executes and update using the first statement
int	<a href="#">getElements</a> ( java.lang.String inputStr) Gets the number of elements, which may be found inside a table
void	<a href="#">leave</a> () Closes the database connection

- **Class DMComparer**

Data Mining comparison module

Constructor Summary	
	<a href="#">DMComparer</a> ( <a href="#">Database</a> db, int applicationTolerance) Constructor with database creation and tolerance parameter for noise processing
Method Summary	
static void	<a href="#">main</a> ( java.lang.String[] args) Main entrace for separate testing only
boolean	<a href="#">ruleApplies</a> ( java.lang.String part1, java.lang.String part2) Compares two parts of strings to see whether - with a certain tolerance - they are similar
boolean	<a href="#">toBeBlocked</a> ( java.lang.String head, java.lang.String data) Decides whether a connection shall be blocked or not
void	<a href="#">updateResultSetNormal</a> () Reads all normal operation cases from the secondary data structure
void	<a href="#">updateResultSetProblem</a> () Reads all problem cases from the secondary data structure

- **Class DMPlugin**

Class for the filtering of raw data do condensed conclusions, i.e., transference between primary and secondary data structure

Constructor Summary	
	<a href="#">DMPlugin</a> ( <a href="#">Database</a> db, int minRulePrecision, int minElementsPerRule) Constructor with reception of parameters
Method Summary	
void	<a href="#">cleanUpConclusion</a> (boolean normal) Cleans up the secondary data structure
void	<a href="#">cleanUpTable</a> (boolean normal) Cleans up the primary data structure
void	<a href="#">flushRules</a> () Resets rule sets in memory
void	<a href="#">jotDownNormalRule</a> (int rule) Puts a rule into the database
void	<a href="#">jotDownProblemRule</a> (int rule) Puts a problem rule into the database
static void	<a href="#">main</a> ( java.lang.String[] args) Side entrance for separate testing only
void	<a href="#">makeParameters</a> ()

	Gets the necessary datasets from the database for analysis
void	<a href="#">makeRules()</a> Main "make" function
void	<a href="#">readIntoSums()</a> Makes the sums of incidents over the primary data structure
void	<a href="#">resetDbPointer</a> (boolean normal) Refreshes the data in the resultsets and resets the database pointer to the first record
void	<a href="#">toNormalRule</a> (java.lang.String head, java.lang.String data) Puts a normal operation rule into temporary memory
void	<a href="#">toProblemRule</a> (java.lang.String head, java.lang.String data) Puts a problem rule into temporary memory
void	<a href="#">transferToDB</a> (boolean normal) Handles rule transfer to the database

- **Class DMSum**

Class that establishes the sums from the primary data structure, needed for the Data Mining algorithm

Constructor Summary	
	<a href="#">DMSum</a> (int minRulePrecision, int minElementsPerRule) Constructor with reception of essential parameters
Method Summary	
void	<a href="#">addElementData</a> (int locale, int element) Adds an element on the data part sums matrix
void	<a href="#">addElementHead</a> (int locale, int element) Adds an element on the head sums matrix
void	<a href="#">dumpAll</a> () Dumps the sums on console - for debug purposes
void	<a href="#">flushSumsData</a> () Initializes the data part sums
void	<a href="#">flushSumsHead</a> () Initializes the head sums
int	<a href="#">getElementData</a> (int locale, int element) Returns an element of the data sums matrix
int	<a href="#">getElementHead</a> (int locale, int element) Returns an element of the head sums matrix
java.lang.String	<a href="#">makeDataRule</a> (boolean problem) Makes a rule from the data part of the sums matrix
java.lang.String	<a href="#">makeHeadingRule</a> (boolean problem) Makes a rule from the head part of the sums matrix
boolean	<a href="#">validRule</a> (java.lang.String text) Examines a rule on validity using the minimum elements parameter

- **Class Feasibility**

Class for the execution of database tests

Constructor Summary	
<a href="#">Feasibility()</a>	Constructor with the creation of parallel database connection
Method Summary	
static void	<a href="#">main</a> ( java.lang.String[] args) Side entrance - there is no link to the main program as this routine is for testing only
void	<a href="#">test1Space</a> () 1st test of space consumption
void	<a href="#">test1SpeedW</a> () 1st test of writing speed
void	<a href="#">test2Space</a> () 2nd test of space consumption
void	<a href="#">test2SpeedW</a> () 2nd test of writing speed
void	<a href="#">test3Space</a> () 3rd test of space consumption
void	<a href="#">test3Speed</a> () 3rd test of writing speed
void	<a href="#">test4Space</a> () 4th test of space consumption
void	<a href="#">test4Speed</a> () 4th test of writing speed
void	<a href="#">test4SpeedR</a> () 4th test of reading speed

- **Class FG**

Main class of the system Fibered Guard, administrator of the main functions

Nested Class Summary	
(package private) class	<a href="#">FG.ConfigListener</a> Listener to config button
(package private) class	<a href="#">FG.ExitListener</a> Listener to exit button
(package private) class	<a href="#">FG.ExitListener2</a> Listener to window exit button
(package private) class	<a href="#">FG.RealLifeListener</a> Listener to real life execution button
(package private) class	<a href="#">FG.SimulationListener</a> Listener to simulation button

Constructor Summary
---------------------

<a href="#">FG()</a>	Constructor with the creation of the window
<b>Method Summary</b>	
void	<a href="#">addActionListeners()</a> Adds actionlisteners to buttons
void	<a href="#">addChangeHistory()</a> Puts the change history into the central text area
void	<a href="#">addTitle()</a> Puts the title info into the central text area
void	<a href="#">addWindowListener()</a> Adds a windowlistener to the main window (to treat closing of the database connection)
void	<a href="#">initObjects()</a> Intializes database and logging object
void	<a href="#">initWindow()</a> Forms the layout of the main window
static void	<a href="#">main</a> (java.lang.String[] args) Main program with creation of object only

- **Class FG.ConfigListener**

Listener to config button

<b>Constructor Summary</b>	
<a href="#">FG.ConfigListener()</a>	
<b>Method Summary</b>	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class FG.ExitListener**

Listener to exit button

<b>Constructor Summary</b>	
<a href="#">FG.ExitListener()</a>	
<b>Method Summary</b>	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class FG.ExitListener2**

Listener to window exit button

<b>Constructor Summary</b>	
<a href="#">FG.ExitListener2()</a>	
<b>Method Summary</b>	

void	<a href="#">windowActivated</a> ( java.awt.event.WindowEvent arg0)
void	<a href="#">windowClosed</a> ( java.awt.event.WindowEvent arg0)
void	<a href="#">windowClosing</a> ( java.awt.event.WindowEvent arg0)
void	<a href="#">windowDeactivated</a> ( java.awt.event.WindowEvent arg0)
void	<a href="#">windowDeiconified</a> ( java.awt.event.WindowEvent arg0)
void	<a href="#">windowIconified</a> ( java.awt.event.WindowEvent arg0)
void	<a href="#">windowOpened</a> ( java.awt.event.WindowEvent arg0)

- **Class FG.RealLifeListener**

Listener to real life execution button

Constructor Summary	
	<a href="#">FG.RealLifeListener</a> ( )
Method Summary	
void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)

- **Class FG.SimulationListener**

Listener to simulation button

Constructor Summary	
	<a href="#">FG.SimulationListener</a> ( )
Method Summary	
void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)

- **Class GeneralConfig**

Class corresponding to the general configuration window of FG

Nested Class Summary	
(package private) class	<a href="#">GeneralConfig.CancelListener</a> Listener to the "cancel" button
(package private) class	<a href="#">GeneralConfig.OkListener</a> Listener to the "ok" button

Constructor Summary	
<a href="#">GeneralConfig</a> ( <a href="#">Database</a> databaseObj, <a href="#">Log</a> logObj)	
Constructor with the creation of the window	
Method Summary	
void	<a href="#">addActionListeners</a> () Adds the actionlisteners to the buttons
void	<a href="#">arrangeWindow</a> () Creation of window layout
void	<a href="#">getValues</a> () Get current configuration values from the database
void	<a href="#">setValues</a> () Writes configured values to the database

- **Class GeneralConfig.CancelListener**

Listener to the "cancel" button

Constructor Summary	
<a href="#">GeneralConfig.CancelListener</a> ()	
Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class GeneralConfig.OkListener**

Listener to the "ok" button

Constructor Summary	
<a href="#">GeneralConfig.OkListener</a> ()	
Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class GridBagUtil2**

Utility class to ease up the use with the GradBagLayout

Nested Class Summary	
(package private) static class	<a href="#">GridBagUtil2.Elemento</a> Interface with GridBagLayout



Constructor Summary	
<a href="#">GridBagUtil2</a> (java.lang.String texto)	Constructor with the creation of the frame
Method Summary	
void <a href="#">add</a> (java.awt.Component comp, int x, int y, int width, int height)	Add an element to the frame
void <a href="#">addExtended</a> (java.awt.Component comp, int x, int y, int width, int height, int fill, int anchor)	Add an element to the frame
void <a href="#">hide</a> ()	Hides the frame
void <a href="#">setSize</a> (int x, int y)	Configures size of the frame
void <a href="#">show</a> ()	Shows the frame on screen

- **Class GridBagUtil2.Elemento**

Interface with GridBagLayout

Constructor Summary	
<a href="#">GridBagUtil2.Elemento</a> ()	
Method Summary	
static void <a href="#">add</a> (java.awt.Container cont, java.awt.Component comp, int x, int y, int width, int height, int weightx, int weighty, int fill, int anchor)	

- **Class IndexException**

Class defining a particular exception on the pool of connections ("no item to release")

Constructor Summary	
<a href="#">IndexException</a> ()	
Methods inherited from class java.lang.Throwable	
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString	

- **Class InterruptMessage**

Warning message handling - for debug purposes

Constructor Summary	
<a href="#">InterruptMessage</a> ()	
Method Summary	

static void	<a href="#">err</a> (java.lang.String inputStr, java.lang.Exception e) Debug messages, which WILL be printed with the stacktrace on console
static void	<a href="#">print</a> (boolean printMe, java.lang.String inputStr) Debug message without display of stack trace

- **Class LFDataStorage**

Data type class for the storage of logical fibering data and its conversion from and to several data types

Constructor Summary	
	<a href="#">LFDataStorage</a> ( ) Creation of the boolean variable and initialization
Method Summary	
void	<a href="#">flush</a> ( ) Initialization of all types of variables (boolean, integer and char)
java.lang.String	<a href="#">getBool</a> ( ) Gets the whole boolean value
int	<a href="#">getBoolAt</a> (int nr) Gets a single binary value of the whole data type
char	<a href="#">getChar</a> ( ) Gets the character value
int	<a href="#">getInt</a> ( ) Gets the integer value
static void	<a href="#">main</a> (java.lang.String[] args) Main routine (side entrance) for testing only
void	<a href="#">setBool</a> (java.lang.String boolStr) Setting of boolean value and automatic conversion to all others
void	<a href="#">setChar</a> (char ch) Setting of character value and conversion to other formats
void	<a href="#">setInt</a> (int inValue) Setting of integer value and automatic conversion to all others

- **Class Limit**

Class for normalization speedup only

Constructor Summary	
	<a href="#">Limit</a> ( )

- **Class Log**

Class for the administration of logging - debug purposes only

Constructor Summary	
	<a href="#">Log</a> ( ) Creation of the log-File and application of the buffered Writer
Method Summary	

void	<a href="#">addToLog</a> (java.lang.String text) Writes a string to the log-File
void	<a href="#">closeLog</a> () Closes the log-File

- **Class MySQL**

Class, which handles the physical connection to the local MySQL database

Constructor Summary	
	<a href="#">MySQL</a> () Constructor, which loads the driver and performs login
Method Summary	
java.sql.Connection	<a href="#">getConnection</a> () Gets the current connection with the MySQL database

- **Class RandomGenerator**

Class used for the generation of random values - used in the simulation of DDoS attacks

Constructor Summary	
	<a href="#">RandomGenerator</a> ()
Method Summary	
static void	<a href="#">main</a> (java.lang.String[] args) Side entrance for separate testing only
int	<a href="#">randomBigInt</a> () Returns a large random integer value (limit configurable)
int	<a href="#">randomInt</a> () Returns a random integer value
java.lang.String	<a href="#">randomIP</a> () Produces a complete random IP address
int	<a href="#">randomIPNr</a> () Produces a random IP number (single octet)
boolean	<a href="#">randomPercent</a> (int probability) Returns a boolean decision based on a provided probability
java.lang.String	<a href="#">randomProgram</a> () Produces a random program name
double	<a href="#">randomSynapse</a> () Produces a random value for synapse initialization (ANN)

- **Class RealExecution**

Nested Class Summary	
(package private) class	<a href="#">RealExecution.Executor</a> Inner class for constant monitoring (thread) and action

Constructor Summary	
<a href="#">RealExecution</a> ( <a href="#">Database</a> databaseObj, <a href="#">Log</a> logObj)	
Constructor with the creation and layout of the window and binding of the listeners	
Method Summary	
void	<a href="#">adjustOperation</a> () Defines the system mode
int	<a href="#">criticalResourceIndex</a> () Returns the critical resource index (for statistics)
java.lang.String	<a href="#">getMode</a> () Gets the system mode
void	<a href="#">hardCodeInit</a> () Hardcoded system parameters
int	<a href="#">overallResourceIndex</a> () Returns the overall resource index (for statistics)
void	<a href="#">softCodeInit</a> () Softcoded system parameters (to be implemented)

- **Class RealExecution.Executor**

Inner class for constant monitoring (thread) and action

Constructor Summary	
<a href="#">RealExecution.Executor</a> ()	
Method Summary	
void	<a href="#">getConnection</a> (int nr) Gets a single connection from those stored in stack memory
void	<a href="#">jotSignatureToDatabase</a> (int nr) Saves a connection to the primary data representation
void	<a href="#">optimizeDB</a> () Deletion of expired database records
void	<a href="#">pollConnections</a> () Gets all current connections and stores them in memory
boolean	<a href="#">preBlocked</a> (int nr) Pre-analysis of a connection - taking ping response into account
void	<a href="#">run</a> () Execution routine of the thread
void	<a href="#">stopRunning</a> () Method to stop the thread
int	<a href="#">toBeBlocked</a> (int nr) Main defense routine
boolean	<a href="#">toBeJotted</a> () Decides whether a connection shall be saved to the database (depending on the system mode)

- **Class SimulatedAttackDefinition**

Class to administrate the simulated attacks (simulation mode)

Nested Class Summary
----------------------

(package private) class	<a href="#"><u>SimulatedAttackDefinition.cancelListener</u></a> Listener on the "cancel"-Button
(package private) class	<a href="#"><u>SimulatedAttackDefinition.changeListener</u></a> Listener on the "change"-Button
(package private) class	<a href="#"><u>SimulatedAttackDefinition.deleteListener</u></a> Listener on the "delete"-Button
(package private) class	<a href="#"><u>SimulatedAttackDefinition.DestinationDNSListener</u></a> Listener on the destination DNS checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.DestinationIPLListener</u></a> Listener on the destination IP checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.downListener</u></a> Listener on the "down"-Button
(package private) class	<a href="#"><u>SimulatedAttackDefinition.Impact1Listener</u></a> Listener on the impact1 checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.Impact2Listener</u></a> Listener on the impact2 checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.Impact3Listener</u></a> Listener on the impact3 checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.insertListener</u></a> Listener on the "insert"-Button
(package private) class	<a href="#"><u>SimulatedAttackDefinition.IterationsListener</u></a> Listener on the iterations checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.newListener</u></a> Listener on the "new"-Button
(package private) class	<a href="#"><u>SimulatedAttackDefinition.ProgramListener</u></a> Listener on the program checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.SourceDNSListener</u></a> Listener on the source DNS checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.SourceIPLListener</u></a> Listener on the source IP checkbox
(package private) class	<a href="#"><u>SimulatedAttackDefinition.upListener</u></a> Listener on the "up"-Button


### Constructor Summary

[SimulatedAttackDefinition](#)([Database](#) databaseObj, [Log](#) logObj)

Constructor with the creation of the window and its layout	
<b>Method Summary</b>	
void	<a href="#">adjustUpDown()</a> Adjusts the "up" and "down" buttons according to the current position in the list
void	<a href="#">clearEntries()</a> Initializes all inputs on screen
void	<a href="#">determineOperation()</a> Determines "insert" or "browse" window operation
void	<a href="#">getCharacteristics()</a> Obtains the system characteristics (names) from the database
void	<a href="#">getData()</a> Gets the data from the resultset and writes it into the window
java.lang.String	<a href="#">getIdForName(java.lang.String inputStr)</a> Gets the id of an effect from its name
java.lang.String	<a href="#">getNameForId(java.lang.String effectNr, java.lang.String inputStr)</a> Returns the name, which is linked to the number of an effect
void	<a href="#">locateID(java.lang.String locateID)</a> Finds an ID on the resultset
void	<a href="#">operationBrowse()</a> Initializes the buttons for "browse" mode
void	<a href="#">operationNew()</a> Initializes the buttons for "insert" mode
void	<a href="#">refreshResultSet()</a> Reloads the resultset (of simulated connections) from the database
java.lang.String	<a href="#">validateEntries()</a> Validates the entries for further processing

- **Class SimulatedExecution**

java.lang.Object

 **SimulatedExecution**

---


```
public class SimulatedExecution
extends java.lang.Object
```

<b>Constructor Summary</b>	
<a href="#">SimulatedExecution(Database databaseObj, Log logObj)</a> Creation of the window, layout and binding of actionlisteners to buttons	
<b>Method Summary</b>	
void	<a href="#">adjustOperation()</a> Adjusts the operation mode
int	<a href="#">criticalResourceIndex()</a> Gets the critical resource index (for statistics)
void	<a href="#">disableUnnecessary()</a> Disables unactive resource slots
void	<a href="#">getDescriptions()</a>

	Load the descriptions of the system characteristics from the database
java.lang.String	<a href="#">getMode()</a> Gets the system operation mode
void	<a href="#">getPercentage()</a> Makes the percentages of resource usage
void	<a href="#">goMode()</a> Configure buttons for "go"
void	<a href="#">hardCodeInit()</a> Hardcoded initialization of system parameters
int	<a href="#">overallResourceIndex()</a> Gets the overall resource index for statistics
void	<a href="#">publishValues()</a> Writes the internally calculated values to the window
void	<a href="#">readCourseFile()</a> Reading of attack course file
void	<a href="#">resetValues()</a> Resets all statistical values on screen
void	<a href="#">setOk(int i)</a> Sets a resource on screen to green color
void	<a href="#">setSevereProblem(int i)</a> Sets a resource on screen to red color
void	<a href="#">setSlightProblem(int i)</a> Sets a resource on screen to orange color
void	<a href="#">softCodeInit()</a> Softcoded initialization of system parameters (to be implemented)
void	<a href="#">stopMode()</a> Configure buttons after "stop"

- **Class SystemCharacteristicDefinition.upListener**

java.lang.Object

 **SystemCharacteristicDefinition.upListener**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

---

```
class SystemCharacteristicDefinition.upListener
extends java.lang.Object
implements java.awt.event.ActionListener
```


Listener on the "up"-Button

Constructor Summary	
<a href="#">SystemCharacteristicDefinition.upListener()</a>	
Method Summary	
void	<a href="#">actionPerformed()</a> (java.awt.event.ActionEvent arg0)

--	--

- **Class `SimulatedExecution.CancelListener`**

java.lang.Object

 `SimulatedExecution.CancelListener`

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SimulatedExecution](#)

```
class SimulatedExecution.CancelListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on "cancel"-Button

**Constructor Summary**

[SimulatedExecution.CancelListener](#)( )

**Method Summary**

void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)
------	--

- **Class `SimulatedExecution.Executor`**

java.lang.Object

 `SimulatedExecution.Executor`

**All Implemented Interfaces:**

java.lang.Runnable

**Enclosing class:**

[SimulatedExecution](#)

```
class SimulatedExecution.Executor
extends java.lang.Object
implements java.lang.Runnable
```

Executor implementing the main thread (operation of the virtual machine)

**Constructor Summary**

[SimulatedExecution.Executor](#)( )

**Method Summary**


void	<a href="#">handleReleases</a> ( ) Handles the releases of connections
void	<a href="#">invokeConnection</a> ( int nr ) Invokes a connection, i.e., "establishes" it with the virtual machine
void	<a href="#">jotSignatureToDatabase</a> ( int nr ) Writes a record from memory into the database



boolean	<a href="#">legitimate</a> (int nr) Returns, whether a connection is legitimate (for statistics only and only possible in simulation!)
void	<a href="#">optimizeDB</a> () Deletes expired records from the database
boolean	<a href="#">preBlocked</a> (int nr) Preanalysis block of connections, using ping response and a configured probability
void	<a href="#">run</a> () Runner of the thread
void	<a href="#">stopRunning</a> () Routine to stop the running thread
int	<a href="#">toBeBlocked</a> (int nr) Main defense routine, which decides whether an attack shall be blocked or not
boolean	<a href="#">toBeJotted</a> () Routine, which - depending on the system mode - decides whether a connection shall be written to the database

- **Class `SimulatedExecution.GoListener`**

java.lang.Object

 `SimulatedExecution.GoListener`

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SimulatedExecution](#)

---

```
class SimulatedExecution.GoListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on "go"-Button

---

**Constructor Summary**


[SimulatedExecution.GoListener](#)()

**Method Summary**

void [actionPerformed](#)(java.awt.event.ActionEvent arg0)

- **Class `SimulatedExecution.ResetListener`**

java.lang.Object

 `SimulatedExecution.ResetListener`

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SimulatedExecution](#)

---


```
class SimulatedExecution.ResetListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on "reset"-Button

Constructor Summary	
	<a href="#">SimulatedExecution.ResetListener()</a>
Method Summary	
void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)

- **Class SimulatedExecution.StopListener**

java.lang.Object

 **SimulatedExecution.StopListener**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SimulatedExecution](#)

---


```
class SimulatedExecution.StopListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on "stop"-Button

Constructor Summary	
	<a href="#">SimulatedExecution.StopListener()</a>
Method Summary	
void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)

- **Class SystemCharacteristicDefinition**

java.lang.Object

 **SystemCharacteristicDefinition**

---

```
public class SystemCharacteristicDefinition
extends java.lang.Object
```

Class for the definition of system characteristics (resources) to be monitored

Nested Class Summary	
(package private) class	<a href="#">SystemCharacteristicDefinition.cancelListener</a> Listener on the "cancel"-Button
(package	<a href="#">SystemCharacteristicDefinition.changeListener</a>

private) class	Listener on the "change"-Button
(package private) class	<a href="#">SystemCharacteristicDefinition.deleteListener</a> Listener on the "delete"-Button
(package private) class	<a href="#">SystemCharacteristicDefinition.downListener</a> Listener on the "down"-button
(package private) class	<a href="#">SystemCharacteristicDefinition.insertListener</a> Listener on the "insert"-Button
(package private) class	<a href="#">SystemCharacteristicDefinition.newListener</a> Listener on the "new"-Button
(package private) class	<a href="#">SystemCharacteristicDefinition.upListener</a> Listener on the "up"-Button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition</a> ( <a href="#">Database</a> databaseObj, <a href="#">Log</a> logObj) Constructor with the creation of the window, definition of layout and binding of listeners
Method Summary	
void	<a href="#">adjustUpDown</a> ( ) Adjusts the buttons "up" and "down" according to the current position in the list
void	<a href="#">clearEntries</a> ( ) Clears all entries on screen
void	<a href="#">determineOperation</a> ( ) Determines "insert" or "browse" mode
void	<a href="#">getData</a> ( ) Reads the data from the resultset into the window
void	<a href="#">locateID</a> ( java.lang.String locateID) Locates a position in the resultset by the id-Value
void	<a href="#">operationBrowse</a> ( ) Sets the operation mode to "browse"
void	<a href="#">operationNew</a> ( ) Sets the operational mode to "insert"
void	<a href="#">refreshResultSet</a> ( ) Refreshes the resultset reading all characteristics from the database
java.lang.String	<a href="#">validateEntries</a> ( ) Validates the input values on screen

- **Class SystemCharacteristicDefinition.cancelListener**

java.lang.Object

 **SystemCharacteristicDefinition.cancelListener**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

---

```
class SystemCharacteristicDefinition.cancelListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on the "cancel"-Button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition.cancelListener()</a>
Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class SystemCharacteristicDefinition.changeListener**

java.lang.Object

 [SystemCharacteristicDefinition.changeListener](#)

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

---

```
class SystemCharacteristicDefinition.changeListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on the "change"-Button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition.changeListener()</a>
Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class SystemCharacteristicDefinition.deleteListener**

java.lang.Object

 [SystemCharacteristicDefinition.deleteListener](#)

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

---

```
class SystemCharacteristicDefinition.deleteListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on the "delete"-Button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition.deleteListener()</a>
Method Summary	
void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)

- **Class SystemCharacteristicDefinition.downListener**

java.lang.Object

 **SystemCharacteristicDefinition.downListener**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

```
class SystemCharacteristicDefinition.downListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on the "down"-button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition.downListener()</a>
Method Summary	
void	<a href="#">actionPerformed</a> ( java.awt.event.ActionEvent arg0)

- **Class SystemCharacteristicDefinition.insertListener**

java.lang.Object

 **SystemCharacteristicDefinition.insertListener**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

```
class SystemCharacteristicDefinition.insertListener
extends java.lang.Object
implements java.awt.event.ActionListener
```


Listener on the "insert"-Button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition.insertListener()</a>
Method Summary	

void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)
------	---

- **Class SystemCharacteristicDefinition.newListener**

java.lang.Object

 [SystemCharacteristicDefinition.newListener](#)

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

**Enclosing class:**

[SystemCharacteristicDefinition](#)

```
class SystemCharacteristicDefinition.newListener
extends java.lang.Object
implements java.awt.event.ActionListener
```

Listener on the "new"-Button

Constructor Summary	
	<a href="#">SystemCharacteristicDefinition.newListener</a> ()
Method Summary	
void	<a href="#">actionPerformed</a> (java.awt.event.ActionEvent arg0)

- **Class TotalConnectionItem**

```
public class TotalConnectionItem
extends ConnectionItem
```

Data structure for an item in the connection pool

Fields inherited from class <a href="#">ConnectionItem</a>	
	<a href="#">attackNr</a> , <a href="#">attackStr</a> , <a href="#">delayNr</a> , <a href="#">delayStr</a> , <a href="#">factorNr</a> , <a href="#">factorStr</a> , <a href="#">SEPARATOR</a> , <a href="#">totalStr</a>
Constructor Summary	
	<a href="#">TotalConnectionItem</a> () Constructor with creation of random generator
Method Summary	
void	<a href="#">addNumToSignature</a> (int num, int limit) Adds a number to the signature string
void	<a href="#">addStopCharData</a> () Adds a stop character to the data part
void	<a href="#">addStopCharHeading</a> () Adds a stop character to the heading
void	<a href="#">addStringToSignatureData</a> (java.lang.String strg, int limit) Creates the signature data part
void	<a href="#">addStringToSignatureHeading</a> (java.lang.String strg, int limit) Creates a signature heading

void	<a href="#"><u>createSignature</u></a> ( ) Creation of total signature - in case of data change, this routine will have to be modified
void	<a href="#"><u>dumpAll</u></a> ( ) Dumps all values on console (debug)
void	<a href="#"><u>dumpSignature</u></a> ( ) Dumps the complete signature to console (debug)
java.lang.String	<a href="#"><u>getData</u></a> ( ) Getter on data part
java.lang.String	<a href="#"><u>getDescription</u></a> ( ) Getter on description
java.lang.String	<a href="#"><u>getDestinationDNS</u></a> ( ) Getter on destination DNS
java.lang.String	<a href="#"><u>getDestinationIP</u></a> ( ) Getter on destination IP
java.lang.String	<a href="#"><u>getDirection</u></a> ( ) Getter on direction
int	<a href="#"><u>getEffect1</u></a> ( ) Getter on effect1
int	<a href="#"><u>getEffect2</u></a> ( ) Getter on effect2
int	<a href="#"><u>getEffect3</u></a> ( ) Getter on effect3
java.lang.String	<a href="#"><u>getHeading</u></a> ( ) Getter on heading
int	<a href="#"><u>getId</u></a> ( ) Getter on Id
int	<a href="#"><u>getImpact1</u></a> ( ) Getter on impact1
int	<a href="#"><u>getImpact2</u></a> ( ) Getter on impact2
int	<a href="#"><u>getImpact3</u></a> ( ) Getter on impact3
int	<a href="#"><u>getIterations</u></a> ( ) Getter on iterations
java.lang.String	<a href="#"><u>getName</u></a> ( ) Getter on name
java.lang.String	<a href="#"><u>getPing</u></a> ( ) Getter on ping
java.lang.String	<a href="#"><u>getProgram</u></a> ( ) Getter on program
java.lang.String	<a href="#"><u>getProtocol</u></a> ( ) Getter on protocol
java.lang.String	<a href="#"><u>getSourceDNS</u></a> ( ) Getter on source DNS
java.lang.String	<a href="#"><u>getSourceIP</u></a> ( ) Getter on source IP
java.lang.String	<a href="#"><u>makeVoidString</u></a> (int limit) Creates a string and initializes it with dots
void	<a href="#"><u>setDescription</u></a> (java.lang.String description) Setter on description

void	<a href="#"><u>setDestinationDNS</u></a> ( java.lang.String destinationDNS) Setter on destination DNS
void	<a href="#"><u>setDestinationIP</u></a> ( java.lang.String destinationIP) Setter on destination IP
void	<a href="#"><u>setDirection</u></a> ( java.lang.String direction) Setter on direction
void	<a href="#"><u>setEffect1</u></a> (int effect1) Setter on effect1
void	<a href="#"><u>setEffect2</u></a> (int effect2) Setter on effect2
void	<a href="#"><u>setEffect3</u></a> (int effect3) Setter on effect3
void	<a href="#"><u>setId</u></a> (int id) Setter on Id
void	<a href="#"><u>setImpact1</u></a> (int impact1) Setter on impact1
void	<a href="#"><u>setImpact2</u></a> (int impact2) Setter on impact2
void	<a href="#"><u>setImpact3</u></a> (int impact3) Setter on impact3
void	<a href="#"><u>setIterations</u></a> (int iterations) Setter on iterations
void	<a href="#"><u>setName</u></a> ( java.lang.String name) Setter on name
void	<a href="#"><u>setPing</u></a> ( java.lang.String ping) Setter on ping
void	<a href="#"><u>setProgram</u></a> ( java.lang.String program) Setter on program
void	<a href="#"><u>setProtocol</u></a> ( java.lang.String protocol) Setter on protocol
void	<a href="#"><u>setSourceDNS</u></a> ( java.lang.String sourceDNS) Setter on source DNS
void	<a href="#"><u>setSourceIP</u></a> ( java.lang.String sourceIP) Setter on source IP
<b>Methods inherited from class <a href="#"><u>ConnectionItem</u></a></b>	
<a href="#"><u>convertToNumbers</u></a> , <a href="#"><u>convertTotalStr</u></a> , <a href="#"><u>getAttackNr</u></a> , <a href="#"><u>getAttackStr</u></a> , <a href="#"><u>getDelayNr</u></a> , <a href="#"><u>getDelayStr</u></a> , <a href="#"><u>getFactorNr</u></a> , <a href="#"><u>getFactorStr</u></a> , <a href="#"><u>getTotalStr</u></a> , <a href="#"><u>setAttackNr</u></a> , <a href="#"><u>setAttackStr</u></a> , <a href="#"><u>setDelayNr</u></a> , <a href="#"><u>setDelayStr</u></a> , <a href="#"><u>setFactorNr</u></a> , <a href="#"><u>setFactorStr</u></a> , <a href="#"><u>setTotalStr</u></a>	



# Curriculum Vitae

30.01.1973	Born in Münster/Westf.  Parents: Prof. DDr. Hans Joachim Schneider Hildegard Schneider (birth name: Schneider)
1979-1983	Primary School "Theresienschule" in Münster
1983-1992	<b>Secondary School "Annette-von-Droste-Hülshoff-Gymnasium" in Münster</b> <b>Conclusion ("Abitur") on 02.06.1992</b>
1992-1993	Community service in Münster
1993-1996	Law Course at the "Westfälische-Wilhelms-Universität" in Münster
1995-1996	Student worker at the Computation Center of the University of Münster
05.05.1996	Emigration to Curitiba (Brazil)
1997-2000	<b>Bachelor Course of Computer Science at the "Centro Universitário Positivo" in Curitiba (Brazil)</b> <b>Conclusion ("B.Sc.") on 31.01.2001</b>
1997-1998	Logistic Assistant at Kvaerner Pulping in Curitiba (Brazil)
1997-2001	Course of Electric Engineering and Pedagogic at the Fernuniversität Hagen
1997-2000	Logistic Technician at Renault do Brasil in Curitiba (Brazil)
2000-2004	System Analyst at gedas do Brasil in Curitiba and São Bernardo do Campo (Brazil)
02.02.2001	Move to Jundiaí (Brazil)
2001-2003	<b>Master Course of Computer Science at the "Pontifícia Universidade Católica de Campinas" in Campinas (Brazil)</b> <b>Conclusion ("M.Sc.") on 17.12.2003</b>
Since November 2004	<b>External Ph.D. student at the "Universität Fridericiana zu Karlsruhe", Institut für Algorithmen und Kognitive System (IAKS)</b>
15.11.2004	Move to Campo Limpo Paulista (Brazil)
2004-2006	System Analyst at Grob do Brasil in São Bernardo do Campo (Brazil)

21.08.2004	Sun Certified Professional
2006	Assistent Professor at the Universidade Paulista in Jundiaí (Brazil) and the Pontifícia Universidade Católica de Campinas, Campinas (Brazil)
Since March 2007	Project Coordinator at Aliança / Hamburg Süd do Brasil in São Paulo (Brazil)

# Publications

## 2006

- M.O. Schneider, J. Calmet: A Logical Fibering Approach to Denial of Service Prevention, In: WSEAS Transactions on Systems , Issue 3, Vol. 6, 2006, pp. 570-575
- M.O. Schneider, J. Calmet: Generic Denial of Service Prevention through a Logical Fibering Algorithm, In: Proceeding of the 5th WSEAS International Conference on Information Security and Privacy (ISP '06), Venice, Italy, 2006, pp. 87-91

## 2005

- M.O. Schneider, J. Calmet: Fibered Guard - A Hybrid Intelligent Approach to Denial of Service Prevention, In: Proceedings of the International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC'05), IEEE Computer Press, Vienna, Austria, 2005, pp. 121-127
- M.O. Schneider, J. Calmet: Denial of Service Prevention through Logical Fibering, In: George E. Lasker, J. Pfalzgraf (Eds.): Advances in Multiagent Systems, Robotics and Cybernetics (Volume I), (Proceeding of the InterSymp'2005), The International Institute for Advanced Studies in System Research and Cybernetics, Tecumseh, Canada, 2005
- M.O. Schneider, J.L.G. Rosa: BioAnt - Biologically Plausible Computer Simulation of an Environment with Ants, In: Proceedings of the International Joint Conference on Neural Networks (IJCNN'05), Montreal, Canada, 2005, pp. 1505-1510

## 2003

- M.O. Schneider, J.L.G. Rosa: Neural Labyrinth Robot - Finding the Best Way in a Connectionist Fashion, In: Anais do IV Encontro Nacional de Inteligência Artificial (ENIA'03), Campinas, Brazil, 2003, pp. 1751-1760
- M.O. Schneider, J.L.G. Rosa: BIOANT - A Biologically Plausible Approach to an Insect Simulation, 1st International Conference on Bioinformatics and Computational Biology (IcoBiCoBi'03), Ribeirão Preto, Brazil, 2003

## 2002

- M.O. Schneider, J.L.G. Rosa: Neural Connect 4 - A Connectionist Approach to the Game, In: Proceedings of the VII Brazilian Symposium on Neuronal Networks (SBRN'02), IEEE Computer Press, Recife, Brazil, 2002, pp. 236-241