Helsinki University of Technology, Networking laboratory
Teknillinen korkeakoulu, Tieteverkkolaboratorio
Espoo 2005 Report 3/2005

On traffic classification and its applications in the Internet

Mika Ilvesmäki

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Electrical and Communications Engineering, for public examination and debate in Auditorium S4 at Helsinki University of Technology (Espoo, Finland) on the 3rd of June, 2005, at 12 o'clock noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking laboratory

Teknillinen Korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Tietoverkkolaboratorio

Distribution:
Helsinki University of Technology
Networking laboratory
P.O.Box 3000
FIN-02015 HUT
Finland

Tel. +358 9 451 2461
Fax +358 9 451 2474

ISBN(print) 951-22-7691-7
ISBN(pdf) 951-22-7692-5
ISSN 1458–0322

Otamedia Oy

Espoo 2005

HELSINKI UNIVERSITY OF TECHNOLOGY
P.O.BOX 1000, FIN-02015 TKK
http://www.hut.fi/

ABSTRACT OF DOCTORAL DISSERTATION

Author: Mika Ilvesmäki

Name of the dissertation
On traffic classification and its applications in the Internet

Date of manuscript May 12th, 2005	Date of the dissertation June 3rd, 2005
☒ Monograph	☐ Article dissertation

Department	Department of Electrical and Communications Engineering
Laboratory	Networking Laboratory
Field of Research	networking, traffic measurements,
Opponent	Professor Giorgio Ventre (Universita‘ di Napoli Federico II)
Pre-examiners:	Professor Jarmo Harju and Professor Hannu Koivisto from Tampere University of Technology, Finland
Supervisor	Professor Raimo Kantola (Helsinki University of Technology)

Abstract

In this work, the methods and applications of traffic classification in the Internet are examined in detail. First, we define and discuss the conceptual environment of traffic classification. We then discuss the performance issues of traffic classification and define a method of visualization to compare the performance of traffic classification implementations.

Previously introduced methods of traffic classification: the static applications, the packet count and the list classifiers are compared with each other. We find these methods to perform quite well when analyzed as performing in an IP router, but to be rather ambiguous as to the effect they cause to the user.

We introduce an implementation of dynamic traffic classification to two classes using learning vector quantization (LVQ) for flow analysis data and find it to perform well in a simulated environment using flow analysis made on traffic measurements. In comparison to the previous methods of traffic classification, we see that the LVQ classifier has adequate performance. We also study a method of traffic classification using consecutive flow analysis with varying values of the parameters of the flow and find that we are able to classify traffic to 2 or 3 different classes. Within the classes the applications are similar in measured behavior and thus may provide help in realizing some advanced Internet service architectures.

Finally, we also observe the application of the dynamic classifier in an Internet router and in the Internet itself. We argue that the implementation of the dynamic classification method is feasible in the network.

Keywords	traffic measurements, traffic classification, Internet, neural networks		
UDC	004.71:004.738.5:621.39	Number of pages	151 pp.
ISBN (printed)	951-22-7691-7	ISBN (pdf)	951-22-7692-5
ISBN (others)	-	ISSN	1458-0322
Publisher	Networking Laboratory / Helsinki University of Technology		
Print distribution			
☒ The dissertation can be read at http://lib.hut.fi/Diss/2005/isbn9512276925/			

TEKNILLINEN KORKEAKOULU PL 1000, 02015 TKK http://www.hut.fi/	VÄITÖSKIRJAN TIIVISTELMÄ
Tekijä: Mika Ilvesmäki	
Väitöskirjan nimi On traffic classification and its applications in the Internet	
Käsikirjoituksen jättämispäivämäärä 11.5.2005	Väitöstilaisuuden ajankohta 3.6.2005
☒ Monografia	☐ Yhdistelmäväitöskirja

Osasto	Sähkö- ja tietoliikenntekniikan osasto
Laboratorio	Tietoverkkolaboratorio
Tutkimusala	Tietoverkkotekniikka
Vastaväittäjä	Professori Giorgio Ventre, Universita‘ di Napoli Federico II, Napoli, Italia
Esitarkastajat	Professori Jarmo Harju ja Professori Hannu Koivisto Tampereen Teknillisestä Yliopistosta
Työn valvoja	Professori Raimo Kantola (Helsinki University of Technology)

Tiivistelmä:

Tässä työssä tarkastellaan Internet-verkkojen liikenteen luokittelua. Aluksi määritellään liikenteen luokittelun käsitteellinen ympäristö ja lisäksi määritellään liikenteen luokittelun eri menetelmien analysointia ja vertailua varten metodi.

Aikaisemmin kirjallisuudessa esiteltyjä liikenteen luokittelumenetelmiä verrataan toisiinsa ja havaitaan, että nämä menetelmät toimivat tehokkaasti laitteiston ja verkon kannalta, kvantitatiiviselta näkökannalta, mutta eivät juurikaan huomioi sitä, millaisten sovellusten tuottamaa liikennettä ne luokittelevat ja siten vaikuttavat yksittäisiin käyttäjiin, kvalitatiiviselta näkökannalta.

Seuraavaksi työssä esitellään uusi menetelmä liikenteen luokitteluun, joka hyödyntää oppivaa vektorikvantisaatiota (LVQ). Tämä liikenteen luokittelumenetelmä toimii tehtyjen simulointien perusteella hyvin sekä kvalitatiivisessa että kvantitatiivisessä mielessä. Uutena piirteenä LVQ-mentelmää käyttävä liikenteen luokitin pystyy luokittelemaan sellaisten sovellusten liikennettä, jotka ovat hyödyllisiä myös verkon käyttäjille. Verrattaessa LVQ luokittimen toimintaa muihin luokittimiin havaitaan, että se toimii useassa tapauksessa yhtä tehokkaasti kuin muutkin vuonluokittelumenetelmät.

Mikäli vuon parametrejä muutetaan, voidaan LVQ luokitinta käyttää myös liikenteen luokittelemiseen useampaan luokkaan. Yhden tällaisen luokan sisällä havaittavat sovellukset ovat luonteeltaan samanlaisia, joten LVQ luokitinta voitaisiin käyttää Internet palveluarkkitehtuurien toteuttamiseen.

Lisäksi työssä tarkastellaan LVQ luokittimen toteuttamiseen liittyviä seikkoja sekä yksittäisessä Internet-reitittimessä että Internet-verkoissa.

Asiasanat	liikennemittaukset, Internet, palvelun laatu, liikenteen luokittelu, neuraaliverkot		
UDK	004.71:004.738.5:621.39	Sivumäärä	151 pp.
ISBN (painettu)	951-22-7691-7	ISBN (pdf)	951-22-7692-5
ISBN (muut)	-	ISSN	1458-0322
Julkaisija	Tietoverkkolaboratorio / Teknillinen Korkeakoulu		
Paonetun väitöskirjan jakelu julkaisijan toimesta.			
☒ Luettavissa verkossa osoitteessa http://lib.hut.fi/Diss/2005/isbn9512276925/			

Acknowledgements

To begin with, to all those not mentioned, thank you, you know who you are.

Naturally, my first thanks go to professor Raimo Kantola whose input and guidance has made the process of completing this thesis an interesting and a rewarding journey. I have also had the privilege to work under professor Kantola's supervision in IPANA-, IMELIO- and IRoNet-projects funded by TEKES and Finnish telecommunication industry. I also want to thank Dr.Tech. Kalevi Kilkki for pointing me to the right direction. Nokia foundation is acknowledged for its generous support in the course of this work. This work has been partly supported by the European Union under the E-Next Project FP6-506869.

Lic.Tech. Marko Luoma is the single most influential person who has participated in and contributed to this work. His insights and ideas on networking, knowledge of details and capability to apply wider perspective both in and outside the field of networking form the cornerstone of this work. Lic.Tech. Markus Peuhkuri has provided me valuable knowledge on network measurements and measurement analysis. Dr.Tech. Jouni Karvo has taught me more on formalizing and structurizing my thoughts than I ever thought possible. These are the guys you want to have close to you when doing thesis work. Thank you!

Ms. Arja Hänninen and all other personnel and colleagues in and outside the networking lab also deserve thanks. The support has been great and is greatly appreciated. In particular, I want to thank the dynamic group of dynamic people who made my lunches and coffee breaks so much more enjoyable both in the lab coffee room and in the exotic and less exotic restaurants of southern Espoo.

Obviously both my mother and father and all of my kin have earned my gratitude for their undying support.

Sensei Yuji Matsuoi and Mr. Jari Renko, respectively my father and my brother in martial arts, have provided me with food for thought and movement for the body in my off-thesis time. Their influence and presence has helped me to keep things in perspective and made it possible for me to physically and mentally survive. Mr. Hannu Väänänen and his family deserve a special thanks for diversion provided when working with the numerous cars of Räpylä Racing. Thanks for keeping the wheels spinning!

Kirsi V. , your support helped me through some of the rough spots during the work. Thank you.

Kirsi W. , thank you for all the love ♡.

Espoo, 2.5.2005,

Mika Ilvesmäki

yksi menee kaikkiin suuntiin
päämääräänsä toinen etsii
jollekin on suunta aivan yhdentekevää
joku kulkee halki ilmojen
joku jalan matkan pölyisen
mut sama taivas on yllä kaikkien

meitä on kaikkialla saman taivaan alla
se taivas on meille jokaiselle yhteinen
ja vaikka värit vaihtuu
valkoisesta mustaan muuttuu
se taivas on kaikille sininen

Pave Maijanen, *Saman taivaan alla*, 2000

Contents

List of Tables

List of Figures

Notation

The following notation is used throughout this work.

$\|$	On the condition that; provided that, if.
$\|\cdot\|$	A function that returns the size of its argument.
$\|\|\cdot\|\|$	A function that instructs to determine the distance value r of its argument. For instance, in a two-dimensional space, $\|\|x-y\|\| = \sqrt{(\|x_1-y_1\|^2+(\|x_2-y_2\|)^2}$.
$\arg(\cdot)$	A function that returns the index of its argument, if available. For instance, $c = arg(m_c)$.
$\max(\cdot)$	A function that returns the maximum value of its argument, if available.
$\min(\cdot)$	A function that returns the minimum value of its argument, if available.

p	A packet.
f	A flow.
a	An application.
c	A traffic class identifier.
s	A classification rule.
m	Mask that defines the flow granularity.
M	Masking function.
τ	Timeout for the flow state to remain active after last packet seen on the flow.
t	Timestamp attribute of a packet.
t_b	Burst length.
$\mathcal{T}$	Time.
r	Distance
$Proto$	The protocol identifier of a packet.
$Saddr$	Source address parameter of a packet.
$Daddr$	Destination address parameter of a packet.
$Sport$	TCP or UDP source port parameter of a packet. Used also as the application identifier in this work.
$Dport$	TCP or UDP destination port parameter of a packet

x	A teaching sample.
α	Learning rate.
v	A codebook vector. Representative of other vectors.
$\mathbb{V}$	A set of codebook vectors.
y	An unclassified measurement sample.
$\mathcal{PC}$	Packet classifier.
$\mathcal{FC}$	Flow classifier.
$\mathcal{TC}$	Traffic classifier.
Π	Set of all packets.
$\mathbb{P}$	A set of packets.
$\mathbb{P}_{\mathbb{T}}$	A set of packets in trace $\mathbb{T}$.
$\mathbb{F}$	A set of active flows.
$\mathbb{D}$	A set of flows to be deleted.
$\mathbb{A}$	A set of applications.
$\mathbb{C}$	A set of traffic classes.
$\mathbb{S}$	The list of policy rules that contain the application identifiers and relevant class information.
$\mathbb{X}$	A teaching set.
B	Flow birthrate factor.
F	Flow count.
S	Flow space factor.
C	Packet classification factor.
A	Application factor.

Chapter 1

Introduction

1.1 Background

One of the basic requirements for future Internet routers is the ability to pinpoint the application flows that require different handling in the network. The limited capability of the traditional IP router architecture to provide different types of service to traffic is becoming an obstacle to network performance improvement. This is because increasingly more applications that need service guarantees from the network emerge while some of the traditional applications used in the Internet might function adequately without any particular service level. Therefore, the requirements of performance and functionality of the future Internet routers are rising to a higher level [2].

Several differing solutions and architectural suggestions exist that introduce service levels and prioritization of traffic to the Internet. Internet service architectures like Differentiated Services and Integrated Services aim to provide varying levels of quality to the critical traffic in the Internet. MPLS[1] aims to provide backup for the service architectures by offering easier and more diverse network management capabilities. At the same time, new architectural concepts enhance the performance of Internet routers through the use of fast packet forwarding with specialized algorithms implemented within the Internet routers.

The growing pressure on the performance of the Internet protocol -based networks requires dealing with several issues such as preserving the overall scalability of the new service architectures, defining the control protocols for routing and distributing information on the identified traffic flows and finally, the main concern in this work, defining flow and traffic identification mechanisms to aid in creating traffic policies [3]. The ability to detect and classify traffic flows from the aggregated traffic mix might result in an added value to this traffic provided we could also offer varying service levels in the network.

The aggregation of traffic flows under similar policy identifiers is a problematic issue since the service level received by a single application flow is difficult, maybe impossible, to determine exactly. Nevertheless, recently this approach where QoS is offered to aggregated traffic flows and where the final decision on classification criteria is left to the network elements, has received a lot of attention in the form of the Differentiated

[1]Multi Protocol Label Switching, see http://www.ietf.org/html.charters/mpls-charter.html

Services approach. Consequently, additional functionality should exist in the network to dynamically determine the policies and policy rules toward different applications.

1.2 The problem

Policies may focus on optimizing the performance of the network components or the focus may be on improving the user's perceived service level. It should be noted that optimizing the performance of the network components does not imply that the user is neglected. On the contrary, a network that functions in an optimal fashion is likely to provide the users increased levels of service. Nevertheless, policy rule creation plays an important role in bringing service levels to the network. The ability to create policies may also bring ability to comprehend and manage the network traffic better. A policy rule, at a very basic level, contains information on the application and the respective priority. The key question for this work, and a question that is rarely given thought, is how one creates the policy rules. An Internet operator may use static choices based on prior knowledge and guesses. Naturally, these guesses may be updated. However, it is not clear on what data should the updates be based. The creation of these rules is not simple.

Since the applications in the Internet are different in their performance requirements, it is necessary to pinpoint the applications needing particular handling, and policy, with accuracy [4]. The assignment of appropriate priority to applications should result in an intuitive and rational set of applications that benefits primarily the user paying for the service. In our point of view, the use of measurements to aid in determining the policy rules is necessary allowing to adapt to varying use of different applications. The purpose of measurement based policy rule creation is to achieve the ability to differentiate traffic into several classes based on measured traffic characteristics. In addition, the automated policy creation should aim to offer users an increase in the measured or perceived performance of the applications creating the traffic. A network operator tries to realize this by using different policies in the network. The rules of the policy determine how different packets of different users are treated in the network.

Since users may not have the expertise to accurately classify their traffic, or, the users may try to misuse the ability to classify the traffic to their unearned advantage, the selection of priority applications might be done in the network by the operator. If the selection of applications, the classification, is done without explicit user intervention, it introduces the problem of how to automatically formulate classification rules so that application traffic may be subsequently classified to higher or lower priority. The essential question we answer in this work is how a set of applications could be identified based on traffic measurements. A problem statement of this thesis follows:

How is it possible to automatically formulate rules, based on measurements of Internet traffic, resulting in an intuitively rational and dynamic application set to be classified for the benefit of the user of these applications, and, at the same time, not to overuse hardware or software resources? Furthermore, if we choose to measure the network, the immediate question asked is "What events in the network should we measure and how should the measurements be analyzed?".

The answers provided in this thesis contain solutions and suggestions on methods of traffic measurement analysis. We will use traffic measurements and analysis thereof as

the means to provide information on what the selection of applications could be based. We do not provide a complete solution and set of analysis methods, but we aim to provide a methodology on how these kinds of solutions may be built and analyzed.

1.3 Overview of the thesis

Because the area of traffic classification is relatively new and has not been extensively studied before, we first look at it in a conceptual level relating it to other components in the packet forwarding process of the Internet router. We then advance to study different traffic classification schemes and proceed to propose a solution of our own for a measurement based traffic classification method. As a side product, we also introduce a method to analyze router performance when using a traffic classification method.

The thesis is organized as follows: In Chapter 2, we discuss the motivation for traffic classification based on previous work. In Chapter 3, we define the basic concepts, such as traffic classification and flow classification and related terms and concepts that are then used throughout this thesis. Also, in Chapter 3, we construct a performance model to enable us to observe the behavior and the performance of the traffic classification methods as they are implemented in IP routers.

In Chapter 4, we take a look at the different methods of traffic classification on a conceptual and functional level. Furthermore, in Chapter 4, we also observe the performance levels of some of the traditional traffic classification approaches by studying traffic classification performance using previously introduced classification methods. We also point out the short-comings in these classification schemes, thus establishing the relevance and existence of the main problem addressed in this thesis.

After examining the performance of some of the existing traffic classification methods we move on to Chapter 5, where we develop the use of the Learning Vector Quantization (LVQ) [5] to analyze particular traffic measurement information. Before that, in Chapter 5, we also introduce flow analysis as a method to characterize Internet applications to aid in forming the classification criteria. We offer a brief look at the LVQ–algorithm and how it is applied to the traffic measurement analysis. We will also look at how the LVQ–classifier may be used to provide information to solve the problem of creating classification rules. We then proceed to develop the architecture and applications of the LVQ –classifier in a network. We study the performance of the 2-class LVQ–classifier and offer a comparison to some of the traditional traffic classification methods.

In Chapter 6 we will extend the method of flow analysis to cover analysis made with varying the value of flow timeout. After establishing the LVQ classifier in Chapter 5 we extend its functionality in Chapter 6 to be able to classify traffic to multiple classes.

Finally, in Chapter 7, we conclude, summarize and present some issues for future research.

1.4 Our contribution

We have studied ways of applying automatic traffic classification methods in the Internet router and in the Internet as a whole. We find the methods of measurement based traffic

classification to be of use as tools for extracting application related information from the network. The major contributions of this work are in the order of appearance:

1. Partly based on previous work, we have constructed a simple methodology to assess the different traffic classification methods from two viewpoints. The first point of view is the use of resources in an IP router in connection oriented and connectionless environments. The second viewpoint is from the viewpoint of the application and the user, and in it we observe the characteristics of the network application profile a particular implementation of a traffic classification scheme detects.

2. We have created a relatively simple method of flow analysis to characterize the Internet applications. Based on these characterizations we perform division of traffic to levels of prioritization, traffic classes. Looking at the results we state that the combined use of packet and flow measurements provides useful information on application characteristics. Observing the behavior of the traffic flows in different timescales and extending the use of flow analysis, we have devised a method to classify Internet traffic to multiple classes.

3. We have developed a measurement based traffic classification method that has notable effects particularly from the user and application point of view. We have applied a specific neural network algorithm, Learning Vector Quantization, to traffic classification. More specifically, with the use of the LVQ–algorithm of supervised learning, we present a solution to the problem of automatically detecting application profiles from the Internet traffic; an issue which has received almost no attention in the previous analysis of traditional traffic classification methods.

The original contributions by the author are found in this and other work as follows:

- In Chapter 3 the formalization of the flow analysis and related concepts as presented in this work have not been published before. Also the context performance model and its visualization are new. However, the context performance model is based on work done by the author and presented in [6] and some of the measured quantities in the performance model are also suggested in [7]. Some background study and motivation for the measurement based policy rule creation using traffic classification has been presented by the author in [8].

- In Chapters 3 and 5 the author has developed the flow analysis methodology. Furthermore in Chapter 5, the author has developed the use and application of the Learning Vector Quantization (LVQ) algorithm to analyze traffic measurement information to produce packet classification rules. The preliminary work on this subject has been published in [9] and it was continued in [10, 11, 12] by the author. However, the terminology and methods have since been refined and space available in the aforementioned publications has not been adequate to present the results in their entirety. Therefore, the work presented in [9, 10, 11, 12] is not comprehensive enough to be included in this thesis and should be considered as preliminary work towards this thesis.

- In Chapter 6 the author has extended the method of flow analysis to include analysis data obtained with different flow timeout values. The extension has been co-developed with Lic.Sc.(Tech.) Marko Luoma. In addition, in Chapter 6, the

author has extended the functionality of the LVQ–classifier to cover the creation of packet classification rules for classifying traffic into multiple classes. The preliminary work on this subject was presented in [12, 13] and we've since continued to revise the methods and terminology on the subject as well as presenting more results. The work presented in [12, 13] does not substantiate the topic of multi-class traffic classification enough and, therefore, should be viewed as preliminary work towards this thesis.

Chapter 2

Motivation for Internet traffic classification

In this chapter we aim to answer two fundamental questions on traffic classification: Why, and Who. The coming chapters will then study the practical question of How. For the question why, we state simple reasons presented in previous work by others and show that, if the current developments for bringing service levels to the Internet continue, traffic classification is justified and needed in the network. Two possible answers exist to the question who should perform traffic classification in the network: the initiative might come from the user or from the network. In particular, if the granularity of the flows is fine it is possible for the user to request a specific service level from the network. If the traffic flows are aggregated and no measurable service may be provided to a single user, it is simpler and makes more sense for the network to form the classification criteria and possibly notify the user of its decisions.

2.1 Introduction

The changes in the Internet traffic characteristics occur in space as the location relative to Internet topology and on the planet in general varies, as well as in time depending on the hour of the day [14, 15, 16]. In short, people use the Internet in various ways all over the world. Internet service providers and network operators need to classify traffic components within their network and evaluate their absolute and relative importance and subsequently create traffic policies to be enforced in the Internet routers [1].

The network operators aim to make profit. The profit earning is based on the operator's business model. Business model is different with different operators depending on their situation in the market. For instance, an ISP, in its startup phase, may concentrate in collecting as many new customers as possible, whereas another ISP may aim to control the growth of the network. The former gets its revenues from customer initiated business contacts (a share of the income brought to an affiliated third party, e.g., Internet store) and the latter produces its cash flow by offering different types of network services to customers. The business models are then realized as different service models (combinations of different service classes) that are eventually sold to customers. These may include free calls to modem pools, offering ADSL-access, the use of certain client applications

(email, www-browser, VoIP-client etc.), offering certain minimum bandwidth, enabling the use of VPN-services etc. The network policy is then created based on the service model resulting in a collection of low-level policy rules. These rules are propagated out to enforcement points using a policy system. In [17] *policy* is defined to be a definite goal, course or method of action to guide and determine present and future decisions. Policies must be enforced to ensure that the users are behaving properly. Policies are implemented or executed within a particular context (such as policies defined within a group of service models). Furthermore, in [18] a policy is defined to be a set of policy rules to administer, manage, and control access to network resources. *Policy rule*, then, is a basic building block of a policy-based system. It is the binding of a set of actions to a set of conditions where the conditions are evaluated to determine whether the actions are performed [18]. This is accomplished via single- or multi-field matching of traffic header and/or payload data with filters. For example, packet filters specify the criteria for distinguishing separable classes of traffic.

Traffic management systems providing for Quality of Service in the Internet consist of a set of high-level rules that are propagated out to enforcement points using a policy system. Figure 2.1 shows a generic policy system. The traffic handlers apply rules handed to them by the decision-making components, which in turn retrieve information they need from data repositories. The rules are propagated out to enforcement points using a policy system. Policy systems as such are straightforward: Policy clients at routers ask the policy parameters from the policy server. Policy servers get the policy data from the information store.

The key question, rarely given much thought, is how does one create the policy rules and the corresponding actions. Should the polices be updated manually based on an expert view, and if so, could we possibly use some statistics based on network measurements to aid in developing the expert view? Furthermore, if we choose to measure the network, the immediate question asked is "What properties should we measure and how should the measurements be analyzed?". Since the applications in the Internet are diversified in their performance requirements, it is necessary to pinpoint the applications needing particular handling, and policy, with accuracy [4].

Our intention, in this work, is to focus on the creation of policy rules based on information gathered from traffic measurements. The policy rules are then evaluated by monitoring the effects of the policy in a single IP router enforcing and using the policy. In this work, we will refer to the process of creating policies based on traffic measurements as traffic classification. Furthermore, we will restrict ourselves to creating policies that are based on combined protocol and TCP/UDP port information on the analyzed traffic.

To achieve good results with traffic classification of this type, the dynamics of Internet traffic must be taken into account when aiming to provide accurate and rational classification results. This is why we use traffic measurements: Ideally, should we choose to divide traffic into different service classes, we would want to obtain and use an application profile containing a set of applications that benefit the user as the traffic of these applications is appropriately classified [19].

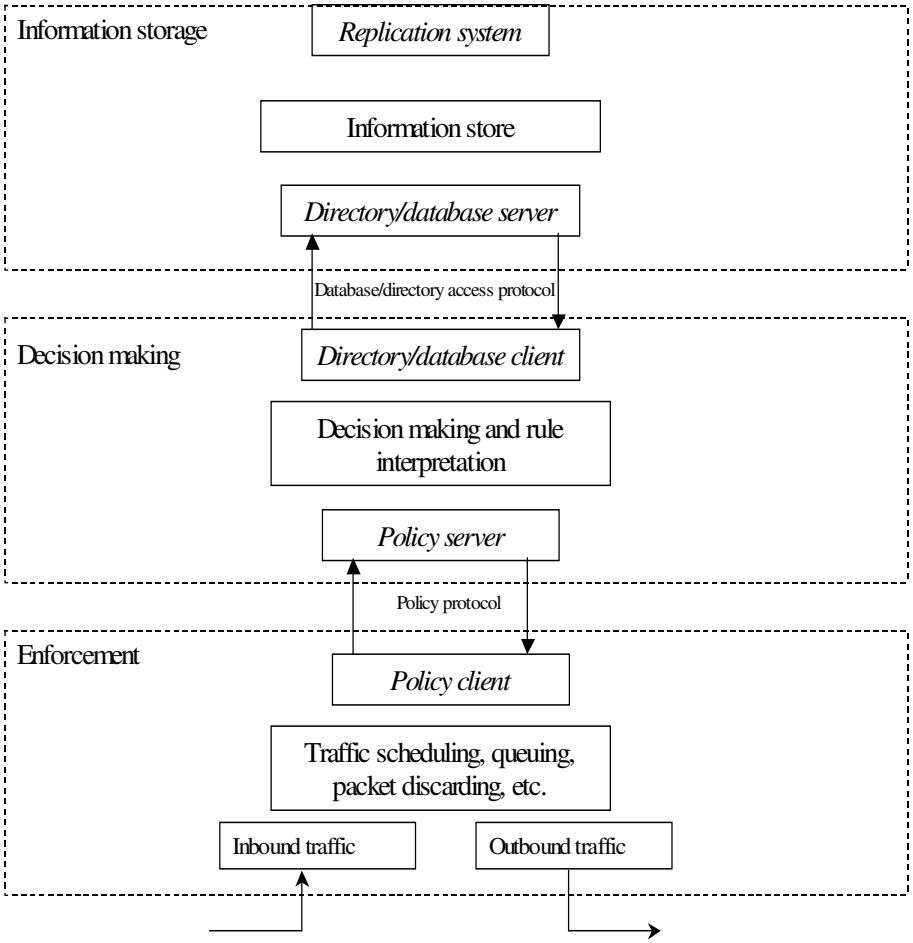


Figure 2.1: A generic policy system [1]

2.2 The need for traffic classification

In the current Internet there is only a single best effort -service class and there exist neither relative nor absolute service classes. Furthermore, it is debatable if we can ever expect the Internet to provide absolute service classes [20, 21, 22, 23]. The traditional data applications, like e-mail and file transfer, are somewhat elastic in their requirements for network performance. These kinds of applications are particularly insensitive to individual packet delays, and typically lack the need for hard real-time constraints. Therefore, the current best-effort-only architecture is ideal for such elastic applications [24].

The emergence of various bandwidth–hungry and delay–sensitive applications, such as video conferencing and voice–over–IP, require, or at least benefit from service classes or some other form of using classification to prioritize packets in the Internet [3]. Also, the use of different p2p–software has begun to consume large amounts of network bandwidth. Therefore, a more varied set of service levels should be offered than just the single class best-effort service and there is a clear need for different levels of service [4]. This need may arise from using applications sensitive to the network performance parameters, or from the general need of users to receive better performance than other users in the network or from the need of the operator to optimize the use of the network resources.

The process of detecting traffic flows is seen as a useful network property [2] for optimizing resource usage in network equipment. The classification of packets and detection of flows is done by matching packet headers with pre-specified filters. The process of determining these filters, the traffic classification process, however, is not addressed in [2]. Shenker argues in [4] that the goal of the network design is to be the answer to the question of how satisfied does the designed network architecture make the users. It is further stated that network performance must not be measured in terms of network-centric quantities like utilization or dropped packets but rather should be evaluated solely in terms of the degree to which network satisfies the service requirements of each user's applications [4]. Therefore, if we were able to detect such traffic that would make better, or even optimal, use of the resources and also appear to the user as rational, we could offer appropriate service levels to the detected parts of the traffic.

It is also stated in [25, 4] that traffic behaving in a particular manner should be classified to the same service class with other traffic behaving in a similar manner. This kind of division is suggested to result in optimal use of network resources and delay any updates on the network infrastructure [25]. From the ISP business point of view this is advantageous, since it helps to maximize the total revenue [26]. Also, according to [26] there is very little added value to know about specific application requirements for the network and, therefore, we divide the traffic into as few classes as possible. Consequently this leads to a partial ISP business statement where the aim is to save money by not having to invest on new network infrastructure and using the old one as efficiently as possible, while offering customers differentiated application treatment to 2 or 3 classes based on measured application behavior. Therefore, the task for the measurement based policy creation would be to provide information that helps to detect and separate application traffic that is behaving in a similar manner.

As an answer to the question why there should be traffic classification, we may state that:

1. There are, and there will be new applications that need varying service level from the network and there seems to be a rising need to classify users or at least the applications they use to different priorities and

2. Traffic classification might provide means to optimize the performance of network elements.

Furthermore, since new applications emerge constantly as old ones fade away and different users use different applications in different parts of the network in different ways, we also require the traffic classification to be dynamic. That is, the method of traffic classification should be able to detect different types of traffic sent by applications.

Therefore, traffic classification should be introduced to the network both to optimize the network element performance and to satisfy the growing needs of performance by the new applications.

2.3 Performing traffic classification in the network

Whether the network or the user should classify a traffic flow is debated quite extensively in [4]:

- Selection of applications by the user means that the user explicitly requests a specific service level from the network. This implies that the network should offer a set of service classes and the applications then indicate to the network which service class they want. In this approach, the incentives needed in the network must encourage users to request the proper service classes for their application. As informal social conventions are seen inadequate to control selfish behavior, the alternative incentive of pricing is brought up. In addition, the network service offerings must be known to applications. After the request for a service level, the network notifies the user if such requested resources exist and reserves these resources.

- The selection and determination of the service level by the network means that the network chooses the applications to be given special treatment and the appropriate service class without explicit information from the user or application. This solution does not require any change to the existing application interface. In addition, the mapping of the application to the service class, and the nature of the service delivered to each service class need not be uniform across all routers nor stable over time, since there is no explicit commitment to a given service level.

2.3.1 User side

Letting the user request traffic classification means user control over the initiation of the resource allocation. User requests at the beginning of his session a service level from the network and the network then decides whether such a service level can be provided. If resources are sufficient the service is then established. However, user requests may be conflicting with each other and fulfilling these requests may cause traffic streams interfering each other if careful on-line traffic control is not utilized. To get around this problem, it may be necessary to give up intentions of providing absolute end-to-end QoS and apply classification in the network so that it defines the applications relative priority against each other.

If the user triggers the requested service level, the user, most likely, wants to also measure the level of service [27]. This suggests that the given service level should be absolute and measurable. This requires that the individual user has his service parameters stored throughout the packet path. Furthermore, this means that every node has to store information on every flow that passes through. This per-flow state may never be practical or feasible in the Internet or at least we can expect that only a very limited set of connections might have their individual states stored in the network. Furthermore, with a user triggered service level there may be situations when the service request can not be fulfilled indicating the need for admission control mechanisms [27]. In the current Internet, the user request method is not applied, although proposals exist to introduce this functionality to the network. One of the major suggestions is the Integrated Services

-architecture [28] to provide per-flow end-to-end service level. The use and analysis of traffic classification methods in this context seems quite useless, since the problems lie more in the areas of bringing together the mechanisms and information provided by admission control, resource allocation and source characterization.

Finally, since there is no isolation between traffic streams created by different users in the Internet, the users can easily disturb each other if they were given complete control of traffic classification. This would not be optimal from the point of view of the whole user population nor from the point of view of the network. It follows that it hardly makes sense to leave the classification decision completely to the user.

2.3.2 The network side

The parameters and rules that govern the classification process and subsequent building of application lists or other classification rules may be determined by the network based on either optimizing the use of resources in a network component or aiming to achieve the best possible user satisfaction. Therefore, the traffic classification methods used by the network are divided into two groups. First, there are methods that primarily intend to optimize the network performance, usually the performance of a single IP router in particular. The applications are chosen as a side result of performing some other related task in the network. For instance, the profile could be formed when working towards forwarding workload reduction using a packet count flow classifier [29, 30, 14, 31, 32]. The detected flows form the application profile. The implicit classification methods primarily take into account only the status and resource restrictions of the network equipment [31]. They do not place any emphasis on the effect the classification is causing to the user. In these methods the target is to optimize the IP router performance and classify traffic flows with a large number of packets. It can be argued, however, that even if the network equipment is functioning in an optimal fashion, the user may not be experiencing any performance improvement or the user may in fact get less than requested from the network in terms of performance. For example, a video stream is assigned to the highest priority blocking all other traffic. This might be optimal in transporting as much data as possible, however, the users' needs are easily overlooked. Second, there are methods that aim to provide the user best possible performance. We could use a specific method to construct the application profile. The method might be based on network managers personal insight to the network and to its traffic, or the list of applications could be constructed based on information obtained and processed from traffic measurements. These classifiers include the classification based on a static application list, an application list based on measurements and classifiers identifying applications based on the packet inter-arrival times [9, 33, 34] or other measured statistics.

Both approaches are valid and implement a certain policy in the network. Our intention is to evaluate the kinds of policies that traffic classification methods may realize or support. We also observe how efficiently the traffic classification methods realize the policies.

If the network does the classification of applications to service classes, there are several problem issues stated in [4]: The network based approach entails a fixed set of application (or service) classes and, therefore, we can not adapt to individual or situational variations within a single application. This is because in the network based approach, the network needs to know something about the requirements of each application. Fur-

thermore, using fixed sets of applications limits the service to those applications [4]. To solve this problem we could monitor, measure and subsequently analyze the traffic, thus being able to provide dynamic sets of applications to service classes using measurement based information. Another problem is that when using the network based classification approach the services offered could be different between different routers [4]. This problem could be overcome by introducing a network or domain wide system of providing information on the service class properties to the individual routers. The severity of this problem is determined in the end by the promises and commitments that a service provider makes regarding the type of service offered to the customer.

Finally, it is suggested in [4] that the service should be requested explicitly by the users or applications with some exceptions, especially when the service levels concern aggregates of traffic flows as, for example, when handing out bandwidth to specific users or application groups. The aggregation of traffic flows under similar policy identifiers is a problematic issue since the service level received by a single application flow is difficult, maybe impossible, to determine exactly. Nevertheless, recently this approach where service classes are offered to aggregated traffic flows and where the final decision on classification criteria is left to the network elements, has received a lot of attention in the form of the Differentiated Services approach [35]. Consequently, additional functionality should exist in the network to dynamically determine and classify the applications that need prioritization. The network functions, that determine the rules and classify the traffic accordingly, are referred to as traffic classification in this work.

2.4 High level view on implementing traffic classification

The measurement based approach for policy creation presented in this work assumes that the measuring process is able to identify and differentiate application behavior as it is measured. This suggests that the process of traffic classification should be done in the edges of the network as close to the originating application as possible to prevent the effect of queuing and multiplexing on the packet arrivals interfering with the original packet arrival pattern. This is also suggested in [1] and is in accordance with the end to end -design principle: End to end -principle states that mechanisms should not be placed in the network if it can be placed at the end node and that the core of the network should provide a general service, not one that is tailored to a specific application [36].

In an IP router there exist software components that examine and analyze the information in the packet header. Those most relevant to this work are the packet and the flow classification procedures. In practical realizations, a packet classifier identifies the packets on a flow by examining packet headers. This requires the use of an either explicit or implicit flow identifier in the packet header. Implicit flow identifier could be a combination of packet fields, and the explicit flow identifier could be a specific field reserved for this purpose in the packet header. The latter approach simplifies the packet classifier whereas the first approach gives more flexibility in determining the packets belonging to a flow. The actual implementation should take into account the various resource oriented restrictions in an actual IP router. This way we obtain an application profile adapting to the resource usage and status in the network. The use of the packet and traffic classification systems in the path of the packet in an Internet router is shown

in Figure 2.2. We note, that the traffic classifier is both a part of the real-time path and

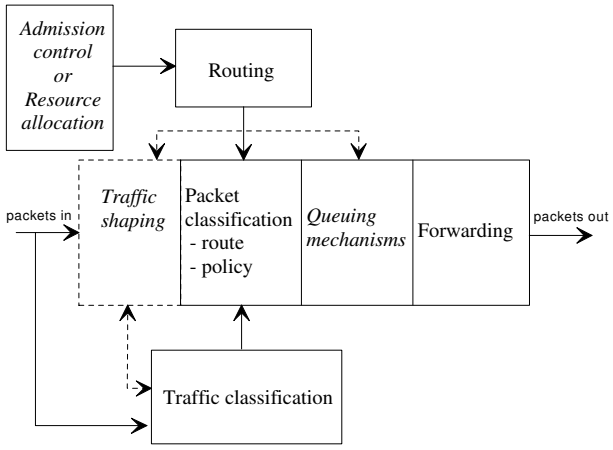


Figure 2.2: Location of the traffic classifier in the Internet router

the non-real time path. Typically, the determination of the application profile would be done off-line whereas the actual application profile would be used together with the packet classification system.

Figure 2.3 describes a network measurement architecture that uses the concept of flow and flow analysis as an aid in obtaining the application profile. Flows, or any other similar

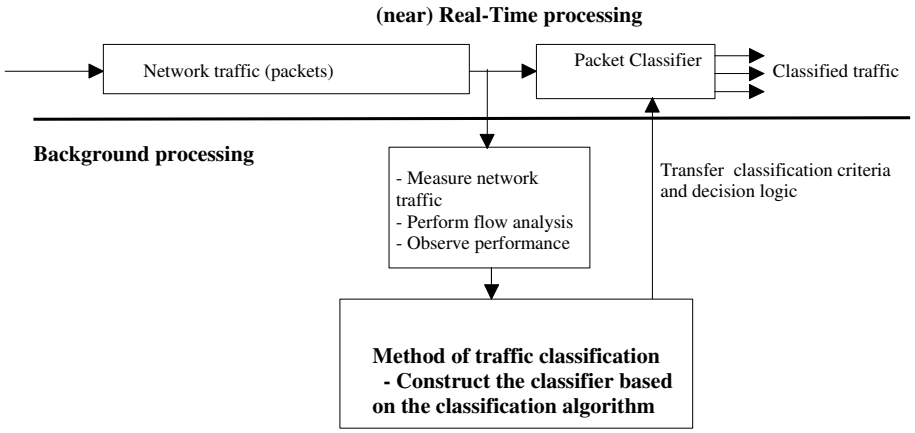


Figure 2.3: A measurement based traffic classification system

concepts, should be used solely on the background when determining the application profile to avoid limiting the applicability of the total scheme to any particular technology or network environment.

To implement a traffic classifier several functional entities need to be devised. These include

- The packet capturing device, which records the timestamp of the packet and relevant header fields to enable packet analysis.

- The flow analyzer to do the actual flow analysis with the possibility to use various definitions for the flow.
- The classifier that is constructed based on traffic measurements and results of measurement analysis added with the possibility to use previously classified data as teaching samples.
- We need decision logic to form the final network application profile, since the classifier may provide several decisions on a single application. The final application profile may then be further distributed to other entities involved in realizing the traffic differentiation in the network.

Another design principle relating to measurement based policy systems is also stated in [36]: Only policy-free, or value-neutral, mechanisms should be designed and allow those who use the system to adjust the mechanisms to match their specific needs. One should not design systems as to dictate the outcome; designs that permit variation are preferred. This means that no preference should be built into the policy creation system. The measures and measurements should dictate the outcome and point out to the policy. The policy creation system and the measurements should be designed and carried out so that the value-neutral -design principle is kept in mind. Therefore, no particular policy should be implied within the design and the outcome of the traffic classification system should be solely based on the choice of flow analysis method, classification algorithm and decision logic.

A suggestion how a policy system using measurement based policy creation should be implemented is presented in Figure 2.4. The proposed architecture in Figure 2.4 aims

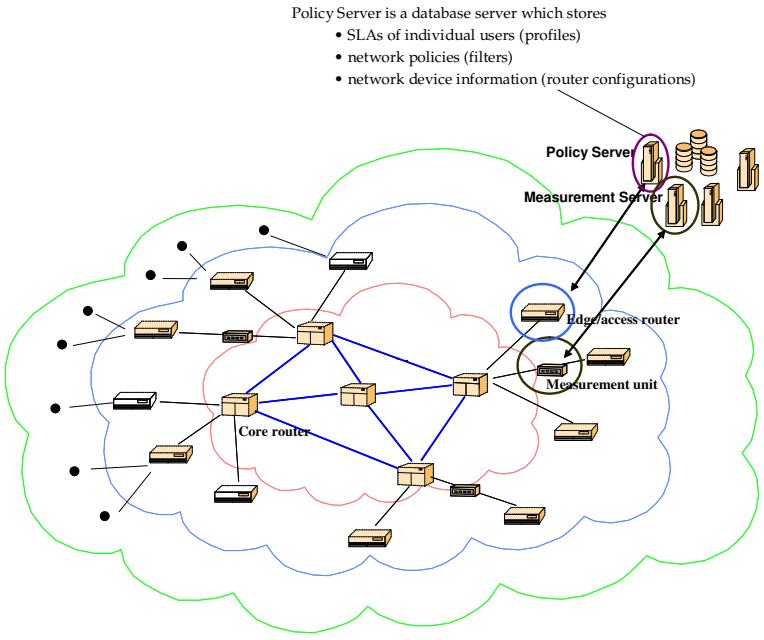


Figure 2.4: Proposed network architecture with measurement based policy creation

to give as much freedom as possible regarding the choice of service architecture, network management style, charging systems and operator business models.

In Figure 2.4 the access routers at the edge of the network have a connection to at least one measurement unit. This unit performs the measurements, does any analysis that is needed on the measurements and based on analysis results determines or suggests the classification and policy rules. These lists are then conveyed to a policy server that decides, based on suggestions sent in by other measurement units, the service level agreements (SLAs) of the customers and other goals set by the network operator, what the policies are in the network. These policies are then distributed to the appropriate customers and edge routers as they connect to the network. The core routers are not involved in determining the policy since they should be dedicated to forwarding packets.

2.5 Summary

This chapter discussed the motivation of traffic classification and debated on whether the network or the user should perform such classification processes. The network-based approach to traffic classification offers at least a stepping stone to implementing service levels in the Internet, without introducing heavy update requirements on user applications and client software. Furthermore, the current trend in bringing service levels to the Internet seem to favor the network based approach. The use of network based traffic classification, however, introduces the problem dealt with in this work: What are the methods to classify traffic flows and how can we differentiate applications from each other?

Chapter 3

Concepts for flow analysis

The intention, in this chapter, is to establish clear definitions and relations of the concepts we will be using later on in this work. First, *traffic classification* is the process of determining the policy rules for packet or flow classification from the traffic measurement analysis data. *Packet classification* is the assignment of priority to the packet in an IP router. *Flow classification* and an *IP flow* are used in the analysis process for traffic measurements to give decision rules to packet classification. The process of traffic classification could be considered a process that remembers the past and resolves collective network traffic properties by observing various statistics in the network and extracting information out of these observations with the aid of various analysis methods.

We will start this chapter with general notes on how traffic measurements should be analyzed. Then we will advance to defining the various concepts and finally we will develop a methodology to assess the various flow classification mechanisms, and to provide us capability to compare the performance of various traffic classification approaches.

3.1 Measuring and analyzing the traffic traces

To begin with, measuring traffic is relatively easy. One needs to capture packets as they arrive on the link (measurement point) and store relevant information on per-packet basis. The idea of using information in the packet headers as the basis for packet classification is extensively studied in [37], where it is seen that by inspecting and properly utilizing the information in the packet headers it is possible to ease the workload of packet processing. The challenge begins as one starts analyzing the packet contents. What should be measured and how should the measurements be analyzed so that the results reflect traffic characteristics? In measuring any phenomenon we may observe the following general events:

- The event itself as it occurs, the count of the events.
- The size or some other quantitative property of the event.
- The frequency (time) between the occurrences of the event.

In the Internet the basic event is the IP packet that is built out of bits and bytes. Measuring packets with the above measures one can quickly see that the measured events relate to counting the packets, measuring its size and observing the temporal behavior with which the packets appear to the measurement point. Because the IP packets contain directional information in the form of the sender and receiver addresses, it is possible to group the packets into smaller sets each containing packets that travel only from one network to another. To be able to differentiate packets sent by different applications we also want to further re-group the packets according to application information. The current network protocols have very few means of detecting individual applications from the packet headers [1]. Advancing beyond TCP/UDP port numbers would take us towards content analysis of the data sent in the packets and require more processing capacity and management efforts in the network (see [38] and references therein for a view on e-mail message analysis). The analysis of packet's (user) data contents is out of the scope of this work. Since the aim is to limit the processing done on a packet, we suggest that any traffic measurement analysis, for policy creation or otherwise, should not advance to use data beyond that provided by IP header and TCP/UDP port numbers.

To be able to group the measured packets, we need a masking function, m, that filters packets according to any (arbitrary) combination of bits in the packet header. Those that make the most sense are the address, protocol and port fields found in the TCP/UDP+IP packet. For instance, if we denote $m(32, 32, 8, 16, 16)$ where each parameter appears in the order they appear in the IPv4-header and TCP/UDP-header and gives the length of the corresponding mask, we use a fine granularity mask that filters out complete fivetuples. Masking combined with some of the basic events can separate IP packets into more interesting sets of packets. For example, should we use the fivetuple masking and introduce a limit to the inter-arrival process, we would arrive with a definition of an IP flow where packets matching the mask m and arriving within a timeout limit τ would belong to the same flow. This type of concept of a flow, or a packet train, was first introduced in [39] and later extensively studied in [40, 41].

Further and iterative use of the masking function provides us with more opportunities to group packets. We could re-apply masking to once masked measurement results and thus obtain more information. For instance, we could define an IP application with mask $m(0, 0, 8, 16, 0)$ and this mask would be applied to a set of flows after fivetuple masking. This would leave us with information (packet count, arrival distributions etc.) grouped by the source port number and protocol identifier. The combination of the source port number and the protocol identifier could then be understood to identify the sending client application and thus the application.

In the Internet the TCP/UDP port numbers enable traffic stream multiplexing from different applications using the same IP address. The port numbers identify an application's interface to use the IP address and may give information on what application is used. We must note that, the use of TCP/UDP port numbers to identify applications do not, however, accurately correlate on the 'interactivity' or any other particular characteristic of the application. Although it is stated in [41] that TCP/UDP ports provide an indication of the expected duration and volume of a flow, many of the new applications utilize the available port number space quite freely and in a random fashion. Network address translators and other gateways of similar nature hide the original port information. Also, the use of generic applications, such as WWW-browsers, may hide the use of distinctly different applications. For instance, loading a streaming video using an appropriate plug-in or common browsing of ordinary web-pages surely produces traffic

streams that are different. All in all, detecting applications using the TCP/UDP port numbers for better service is somewhat problematic.

It should be noted that other combinations of masks and basic measurements might provide interesting possibilities for traffic characterization. As we advance from measuring the count of events, we encounter measures like the event size (in bytes) and duration (flow length in seconds, for instance). These measures vary from packet to packet and from flow to flow which means that we must collect these measures for every packet and flow and form a distribution of the measurement results. We face the same requirement when we measure the occurrence of the event (inter-arrival-times). These distributions probably contain typical characteristics of the underlying application that creates the events. Collecting the traffic measurements would allow us to calculate the means, variances, or higher-order statistics out of these measurements. We could also collect information to determine data distribution of the measurements to help us to further understand the nature of the measured traffic. However, characterizing and parameterizing the distributions is a complex task and resorting to use simple statistical measurements of the distributions, such as means and variances, provides us with very little knowledge on different application characteristics and inevitable loss of information on the distributions themselves. Therefore, we choose, in this work, to measure only the packet and flow counts per application in our search for distinct application traffic characteristics. The choice of the two scalars to represent the application characteristics is made to keep the analysis simple and feasible. Should these measurements be correctly analyzed, we hope that the results provide information on application characteristics and thus aid in differentiating traffic.

3.1.1 Characterizing applications with packet and flow count

The packet count of an application indicates the presence of the application in a network. The flow count gives an indication how bursty and fragmented the sending of the packets by the application is. If we compare the flow count of an application to the total number of flows in the trace, a high relative flow count indicates that the application sends its packets in short bursts, $t_b < \tau$, and that the intervals between the bursts are longer than the flow timeout τ. On the other hand, a low relative flow count indicates longer bursts, $t_b \geq \tau$, or streams, and relatively short intervals between the bursts or streams. However, the flow count does not give a clear view on how much packets the application has sent. If we were to combine the observations on packet and flow count, the application types shown in Table 3.1 could be defined. This would allow us to analyze the behavior of an application simultaneously by the count of packets and flows it creates.

Since the packet count of an application is the same regardless of the flow parameters, we may also observe the development of flow count as a function of the timeout to further characterize the applications. We note that different flow timeout values may change the amount of flows observed per application radically. The shorter the timeout value is, the fewer the number of TCP connections or packets in a UDP stream that can be mapped under one flow. The longer the timeout of the flow is, the more TCP connections or packets of UDP streams aggregate into a single flow.

If the change in flow count as a function of the timeout τ is significant, this indicates that the application sends long bursts or streams ($> \tau$) with short intervals ($< \tau$) between

Table 3.1: Assumed packet and flow characteristics related to application type

Packet and flow characteristics		
Packets	Flows	Property
$\frac{pkt_{app}}{\Sigma pkt} \uparrow$	$\frac{flw_{app}}{\Sigma flw} \downarrow$	High presence, smooth traffic, low in bursts
$\frac{pkt_{app}}{\Sigma pkt} \uparrow$	$\frac{flw_{app}}{\Sigma flw} \uparrow$	High presence, fragmented and bursty traffic
$\frac{pkt_{app}}{\Sigma pkt} \downarrow$	$\frac{flw_{app}}{\Sigma flw} \uparrow$	Low presence, bursty traffic
$\frac{pkt_{app}}{\Sigma pkt} \downarrow$	$\frac{flw_{app}}{\Sigma flw} \downarrow$	Low presence, smooth traffic

sendings. As we reduce τ the flow count increases. On the other hand, if the change in the flow count as a function of the timeout τ is insignificant this indicates that the application sends its packets as short data bursts ($< \tau$) with relatively long intervals ($> \tau$) between sendings. In this case, if we reduce τ, it will not have any significant effect on the flow count.

The discussion and heuristics here will be further backed up by measurement results presented in the beginnings of Chapters 5 and 6. Next, we will proceed in defining the concepts that we will use in the analysis of the measurement results.

3.2 Notation for traffic and packet classification

An individual packet has six attributes that define the packet as a unique sixtuple:

$$p =< t, Saddr, Daddr, Proto, Sport, Dport > \tag{3.1}$$

where t is the timestamp of the packet, *Saddr* and *Daddr* are the source and destination IP addresses, *Proto* is the carried protocol type within the IP packet, and *Sport* and *Dport* are the source and destination TCP or UDP port numbers. All of these properties have to be present or there exists no packet as far as this work is concerned.

We denote by $\mathbb{P}$ any set of consecutive packets in a traffic trace and by $\mathbb{P}_{\mathbb{T}}$ the set of packets in trace $\mathbb{T}$. We denote a packet in $\mathbb{P}$ with p. In addition, we define the superset Π to contain all of the past, current and future packets.

An IP application a, in this work, is defined by the pair made up of transport layer protocol (TCP/UDP) identifier and the source port number, *Proto* and *Sport* in the IP packet p. Therefore, an application can be formalized as a set of packets ($p \in a$) with common *Proto* and *Sport*. We denote by $\mathbb{A}$ the set of all applications related to a trace:

$$\mathbb{A} = \{a \mid \exists p \in \mathbb{P}_{\mathbb{T}} \text{ such that } (p \in a, a = (Sport, Proto))\} \tag{3.2}$$

If the definition of the application would also take into account the destination port number we would experience an abundance of identical applications since the destination port numbers are used in various ways depending on the implementation of the receiving TCP/UDP client. In the future, one could also use IPv6 flow labels, IPv4 ToS/Precedence/DSCP -values [42, 43, 35] or some other means to ensure that the sending application is distinctly identified in the packet headers. This type of application

definition has also been used in relation to performance analysis when IP traffic is carried over switched networks [44].

Owing to the limited size, 16 bits, of the port number field according to the TCP and UDP specifications, the possible values of the application identifier are limited between 0 and 65535. Therefore, the maximum number of applications in Π is 65536 for one protocol *Proto*.

3.3 Definitions for traffic classification

A traffic class c is a member of the set of traffic classes $\mathbb{C}$.

$$\mathbb{C} = \{c\} \tag{3.3}$$

A traffic class identifier c indicates packets that share a common property. A packet classification function assigns the class identity to packets. All packets in a class should be treated equally in the network. Using c we disentangle the application running from the service required, thus negating the controversy between what applications users can run and what service levels an ISP chooses to offer [36].

For the purpose of this work, $\mathbb{S}$ is defined as a list of application identifiers a combined with the information of the corresponding traffic class c. Formally, we can express this in the following manner: The set of classification rules $\mathbb{S}$ consists of individual rules s that are made out of an application identifier a combined with information on the corresponding traffic class c.

$$\mathbb{S} = \{(a, c) | c \in \mathbb{C}, a \in \mathbb{A}\} \tag{3.4}$$

The rule lists may be updated either manually by the network manager or automatically by some method that monitors the network and learns the applications used and their characteristics in the network.

We continue to define a traffic classifier: Traffic classifier, $\mathcal{TC}$, is a function that takes as input selected characteristics of a set of packets, $\mathbb{P}$. As output, $\mathcal{TC}$ produces the set of policy rules $\mathbb{S}$, that may then be used by $\mathcal{PC}$ to assign individual packets p to a traffic class, c as appropriate. More formally,

$$\mathcal{TC}(\mathbb{P}) = \mathbb{S} \tag{3.5}$$

The packet classifier is defined as follows: Packet classifier, $\mathcal{PC}$, is a function that takes as its input the individual packet $p \in \mathbb{P}$ and classification rules $\mathbb{S}$ and produces as output, in the context of this work, the corresponding traffic class c. It is able to distinguish a packet from another using the packet parameters and subsequently classify the packet according to externally determined classification rules $\mathbb{S}$.

$$\mathcal{PC}(p, \mathbb{S}) = c \tag{3.6}$$

Now, we denote a set of packets that belong to a class c with $\mathbb{P}_c$. Packet classification is implemented with packet filters in an IP router. The packet filters parse a portion

of the packet header to find out the packet parameters before forwarding decisions are made. The parsing is based on the set of rules $\mathbb{S}$, defined by the traffic classification process, e.g., network management software, real-time reservation protocols or any other method of traffic classification. The packet classification process is extremely time-critical since it is necessary to do the classification at wire-speed. Packet classification may be implemented using enhanced algorithms designed for this purpose [45, 2, 46, 47, 48, 49].

3.4 Notation and definitions for flow classification and flow analysis

Originally, the concept of flow classification comes from the world of IP switching [29, 32], where it was first used to separate flows from each other. The concept of a flow, or a packet train, was first introduced in [39] and later studied in [40, 41]. In flow classification the rules, that are passed on to packet classification, are initiated based either on some identifier in the packet (IP protocol field values, IP Type of Service bits, TCP/UDP ports) or after a certain threshold is reached, for instance a certain number of packets have arrived on a candidate flow. Flow classifiers typically combine several rules in order to identify the flow.

The following introduces two concepts: flow analysis and flow classification. *Flow analysis* is a method to analyze flow (and packet) counts of an application ($a \in \mathbb{A}$). It returns the number of one-way flows of an application. However, since the use of any application is a two-way process, in *flow classification* the flows are set up regardless of whether the application identifier, and the port number in particular, exists in the source or destination port fields of the TCP/UDP -header. This is to ensure that any performance figures are as realistic as possible. In real networks this corresponds to a situation where the packets containing the requests for particular objects, the actual object transfer and related acknowledgements would receive the same service level from the network. For instance, in flow classification if the rule set contains port number 80 with TCP-protocol, priority is assigned to all packets using TCP that have port number 80 either in their *Sport* or *Dport* field. As far as statistics in this work are concerned, the packet and flow data gathered in flow classification are associated by the packet's source port number.

An *IP flow* is denoted by f. A flow is a set of packets that are travelling from a source to a destination as defined by the parameters of flow timeout τ, and flow granularity using mask m. A flow classifier is defined as a function that returns the set $\mathbb{F}$ of active flows at the end of the observation period, as the flow parameters τ, m and a set of packets are given as input:

$$\mathcal{FC}(\tau, m, \mathbb{P}) = \mathbb{F} \tag{3.7}$$

The flow granularity level m defines the aggregation of individual flows by using a mask for $Saddr, Daddr, Proto, Sport$ and $Dport$. The flow timeout value τ is the limiting time between two consecutive packets to be associated with the same flow. As packets enter the network, they are spaced arbitrarily in time by the applications creating them. If the spacing exceeds the flow timeout value, the following packets, otherwise classified to the same flow, are assigned to a different flow. The parameters τ and m are the production attributes of a flow. These attributes are used for flow identification.

The concept of active flows defined in 3.7 is useful because it allows talking about the

largest number of simultaneous flows and about the total number of flows in a trace using the maximum function (max) and union ($\cup$) respectively as we will see later in this chapter.

A flow $f \in \mathbb{F}$ is then a set of packets that are chosen using τ and m. M is the masking function that takes as input the mask, or granularity m, and the packet. In more specific terms:

$$f = \{p \in \mathbb{P} \mid \exists p' \in \mathbb{P} \text{ such that } (t - t' \leq \tau) \& (M(m,p) = M(m,p'))\} \tag{3.8}$$

We note that a flow as defined in eq. 3.8 contains at least one packet when $p = p'$. We will assume that a flow has a unique identity in a trace. This assumption allows us to cut the trace into shorter subtraces and concatenate the subtraces back together to form the original trace. A flow has other properties in addition to τ and m such as duration, packet arrival distribution and size (in bytes). These properties may be modelled by their respective attributes.

3.4.1 Defining and implementing flow analysis

We note that although the use of any application is a two-way process, the traffic characteristics are in some cases highly asymmetrical. For instance, a long flow of packets may be invoked with just a few packets containing a request for a large data object. Therefore, to be able to accurately characterize applications, we collect all the statistics to be used in flow analysis based on source port *Sport*.

Flow analysis is the method we will be using on a packet trace to gather characterizing information on the behavior of different applications. The data that we collect includes two components: the relative packet and flow counts. These are defined as follows: The normalized classification factor (C_{norm}) for application a', is determined by dividing the individual application packet count with the total number of packets present:

$$C_{norm}(a) = \frac{|\{p \in \mathbb{P} \mid (p \in a)\}|}{|\mathbb{P}|}, \text{ where } \mathbb{P} \subset \mathbb{P}_{\mathbb{T}} \tag{3.9}$$

Respectively, for flow count we measure and determine the normalized number of flows for each application $a \in \mathbb{A}$ as follows:

$$F_{norm}(a) = \frac{|\bigcup_{\mathcal{I}} \mathcal{FC}(\tau, m, \mathbb{P}_{\mathbb{T}}(a))|}{|\bigcup_{\mathcal{I}} \mathcal{FC}(\tau, m, \mathbb{P}_{\mathbb{T}})|} \tag{3.10}$$

It should be noted that the packet and flow count of individual applications are added up respectively to form two aggregated statistics for the particular application, that is, the aggregated packet and flow counts. This way, the occasional exception in the behavior of the application is dampened.

The pseudo-code for implementing flow analysis is shown in Table 3.2. The flows are set up only when packet's *Sport* and *Proto* match the rule entry in $\mathbb{S}$ thus emphasizing the identification of application flows themselves. We also show the gathering of the statistics for further flow analysis. The statistics returned are the packet and flow counts per each application identifier *Sport* and *Proto*. The number of applications is found out by counting *Sport* and *Proto* with non-zero flow count.

Table 3.2: Pseudo code for implementing flow analysis

process Init
$\mathbb{F}', \mathbb{D} = \emptyset$;
$C_{norm}(\mathbb{A}) = 0; F_{norm}(\mathbb{A}) = 0$;
$m = (32, 32, 8, 16, 16)$;
end Init;

process Flow_Analysis
packet p arrives→
$\forall a = (Sport, Proto) \in \mathbb{A}$;
$\mathbb{F} = \mathbb{F}'; C(a') = C(a') + 1$;
if $\exists f \in \mathbb{F}$ `such that`
$p_f \in f$ `and` $t - t_f < \tau$ `and` $M(m,p) = M(m,p_f)$;
then
$\mathbb{F}' = \mathbb{F} \setminus \mathbb{D}$;
$t_f = t$;
else
$\mathbb{F}' = (\mathbb{F} \cup f_p) \setminus \mathbb{D}$;
$F(a') = F(a') + 1$;
$t_f = t$;
fi;

do
$\mathbb{D} = \{f \mid t - t_f \geq \tau\}$;
od;
end Flow_Analysis;
process Stats
$\forall a \in \mathbb{S}$;
Return $C_{norm}(a') = \frac{C(a')}{\sum_{\forall a \in \mathbb{A}} C(a)}$;
Return $F_{norm}(a') = \frac{F(a')}{\sum_{\forall a \in \mathbb{A}} F(a)}$;
end Stats;

3.4.2 Implementing flow classification

The high level pseudo code implementation showing the functional entities of a flow classifier is shown in Table 3.3. The flow classification procedure includes the timeout monitoring of all flows. The flows that time out are moved to a set of deleted flows $\mathbb{D}$. The other option would be to implement a flow state teardown based on, for instance, the teardown procedure of TCP connection. However, this would leave many flows and applications hanging since there are applications that use the UDP protocol.

It should also be noted that the classification rules $\mathbb{S}$ are defined by the traffic classification process and the flow classification procedure is initiated only if these rules exist.

Table 3.3: Pseudo code for implementing a flow classifier

```
process Init
                                    F', D = ∅;
                                    A = {a|(a, c) ∈ S};
                                    m = (32, 32, 8, 16, 16);
end Init;

process Flow_Classification
packet p arrives→
if (a = (Sport, Proto) ∈ A or
a = (Dport, Proto) ∈ A);
then
                                    F = F';
                                    if ∃f ∈ F such that
                                    p_f ∈ f and t − t_f < τ and M(m, p) = M(m, p_f);
                                    then
                                    F' = F \ D;
                                    t_f = t;
                                    else
                                    F' = (F ∪ f_p) \ D;
                                    t_f = t;
                                    fi;
fi;
do
                                    D = {f | t − t_f ≥ τ};
od;
end Flow_Classication;
```

3.5 Evaluation of traffic classification methods

3.5.1 Introduction

After measurement based policy rule creation systems have produced the classification rules, the outcome should be evaluated. A number of things may be considered for evaluation. The interesting question is how well the classifier succeeds in classifying traffic in the network. Are correct rules and policies being produced as defined by the measurement analysis? Is the measurement analysis able to detect traffic characteristics from the measurements? Is the policy, from its own point of view, producing a network that is used more optimally and are the customers satisfied with the way that the policy is treating them? Is the traffic classifier general enough to be used in other network environments without any additional modifications to its parameters?

However, these tasks are relatively difficult to tackle with simple metrics, since the overall success of a classifier is hard to model in an accurate and simple way. Therefore, we have developed a set of indirect metrics that have been derived by applying the classification results in a simple simulated network node environment, and by examining the classification results with simple statistics and subjective evaluation of the classified application lists. With these metrics we hope to be able to evaluate different traffic classification methods.

In this work we focus on evaluating the classification effects to the network and to the user. We divide the analysis to evaluating the content of the policy lists and evaluating the context performance. By context performance we mean evaluating the performance of the environment where the policies are used. We start by developing a method for the context performance evaluation, where the evaluation of the use of the policy lists is done based on simple IP router performance metrics. After presenting the method for context performance evaluation we develop a methodology to evaluate the content of the policy rule lists.

3.5.2 Evaluation of context performance

Context performance characterizes the impact of classification on the network node performance. First we define the metrics we will use for evaluating the context performance. We start by examining the restrictions for a router working in an environment which acknowledges flows or connections [7]: *Processor capacity*, in general, is limiting both the flow birthrate speed as well as the packet forwarding capabilities. *Connection or state space* is a limited resource and used quite liberally when fine–grained flow granularity is applied. Usage of connection or state space could be controlled using the flow birthrate and the flow deletion speed as indicators whether the usage is increasing or decreasing. In a connectionless environment connection space could be understood as using different kinds of headers which implied a better priority handling in the network for flows [50, 51, 52]. In general terms, connection space could also be understood as available flow space, or the amount of resources available for storing flow state information. Depending on the environment this could be understood as connections or entries in a flow database. *Bandwidth, or more accurately, the transmission rate* is also a limited resource; it may not be used without restrictions. One solution would be to add more bandwidth as the traffic increases. This would, however, be only a temporary relief, since new applications and users will be anxious to lay their hands on any free bandwidth they see available. There needs to exist a mechanism that allocates the transmission capacity to connections in a rational fashion. *Delay to setup connection or state* means that the methods of packet classification and subsequent packet action should be simple and not time–consuming [53]. Otherwise, the majority of packets for which the connection or state is established, might already have been sent. In this light, it is critical that any connection candidates detected would receive the decision to establish the connection or the state as quickly as possible [53].

To observe the context performance of a traffic classification scheme we introduce four different factors of context performance that are observed. We call the result the context performance evaluation model. This model evaluates the performance based on a reference point. The reference point is formed by classifying all flows to priority after the first packet on the flow is observed. This performance sets the reference point to which all other classifiers will be compared.

The factors in the context performance evaluation model have been chosen to be as general as possible while providing information for evaluating the traffic classification schemes in different network environments. The chosen factors are based on those presented in [7, 54]. These statistics form the contextual aspects of the performance observations of different traffic classification schemes: *Maximum flow birthrate factor* indicates the birthrate of flows with a particular classifier in relation to the reference classifier that has the highest possible birthrate in a given trace. This factor disregards the actual

cost of setting up flow state, or in some other way utilizing the information of this new flow, since this is an implementation specific issue and, therefore, not in the scope of this work. However, by observing the birthrate maximum we get a notion of the required CPU use for flow state setups. The maximum flow birthrate B for a traffic class c in trace $\mathbb{T}$ is determined by comparing the maximum amount of new flows detected within the appropriate traffic class c in a period of one second ($\Delta\mathcal{T} = 1 < \tau$) to the maximum number of flows detected in a period of one second with all flows classified (reference point):

$$B_c = \frac{\max_{\mathcal{T}} |\bigcup_{\mathcal{T}+\Delta\mathcal{T}} \mathcal{FC}(\tau, m, \mathbb{P}_c) \setminus \bigcup_{\mathcal{T}} \mathcal{FC}(\tau, m, \mathbb{P}_c)|}{\max_{\mathcal{T}} |\bigcup_{\mathcal{T}+\Delta\mathcal{T}} \mathcal{FC}(\tau, m, \mathbb{P}) \setminus \bigcup_{\mathcal{T}} \mathcal{FC}(\tau, m, \mathbb{P})|}, \ \mathbb{P}, \mathbb{P}_c \subset \mathbb{P}_\mathbb{T}, \text{ and } \tau > \Delta\mathcal{T} = 1 \quad (3.11)$$

Results will also present the time $\mathcal{T}$ that starts the (one second) period when the maximum flow setup speed is observed.

Flow space factor shows the maximum number of simultaneous connections in the network for given connection parameters. This factor is important if we wish to give user flows individual connections and the available connection space is limited. It also gives an indication on the memory needs for a flow state system. The flow space factor S in a traffic class c is determined by comparing at any given time the maximum use of flow state space within the appropriate traffic class using the appropriate traffic and flow classification functions to the maximum use of the flow state space when classifying all flows (reference point):

$$S_c = \frac{\max_{\mathcal{T}} |\mathcal{FC}(\tau, m, \mathbb{P}_c)|}{\max_{\mathcal{T}} |\mathcal{FC}(\tau, m, \mathbb{P})|}, \text{ where } \mathbb{P}, \mathbb{P}_c \subset \mathbb{P}_\mathbb{T} \quad (3.12)$$

Classification factor C indicates how many packets of all the packets are classified to class c. The classification is determined by comparing the amount of packets classified to a certain traffic class c using the appropriate classifier with the total packet count in seen in the trace indicated by $|\mathbb{P}_\mathbb{T}|$:

$$C_c = \frac{|\mathbb{P}_c|}{|\mathbb{P}_\mathbb{T}|} \quad (3.13)$$

Application factor indicates how many different applications are present in the trace according to the application definition in the beginning of Chapter 3. This factor is dependent on the definition of the application and therefore varies if the definition is altered. This factor tells us how large a portion of the applications are classified to a particular traffic class. This factor may indicate any imbalance in differentiating traffic to classes and thus suggest additional access restrictions to a traffic class. The application factor A in a traffic class c is determined by comparing the number of applications in the traffic class c to the total number of applications in the trace (as determined when all flows are classified):

$$A_c = \frac{|\{a \mid \exists p \in a, (a, c) \in \mathbb{S}\}|}{|\mathbb{A}|} \quad (3.14)$$

The first two factors, the flow birthrate factor and the flow space factor, are closely related to situations where the network needs to store information on the flows. These two factors lose their relevance if the network functions in a fashion where no status information needs to be updated or stored.

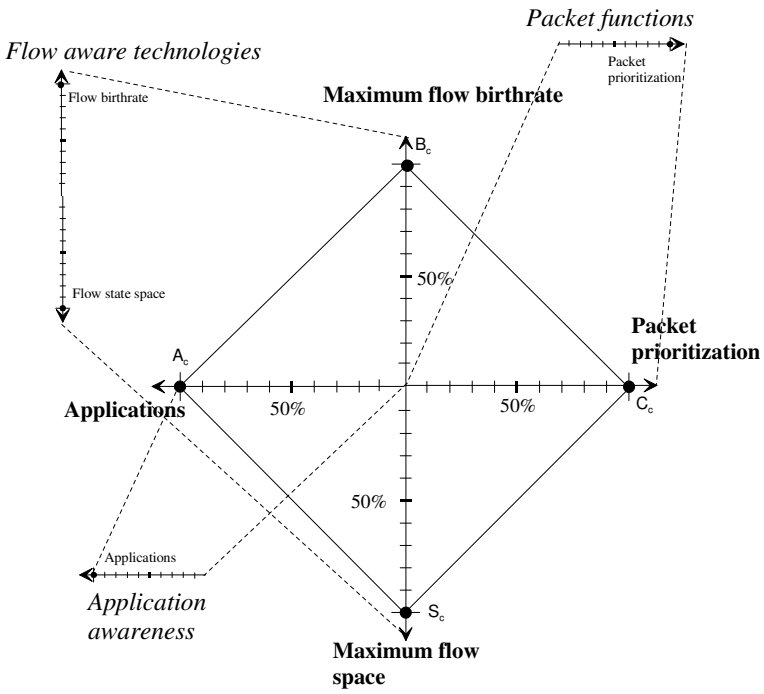


Figure 3.1: The context performance model for comparing traffic classification methods

To visualize the four context performance factors we introduce the quadrilateral model for comparing different traffic classification schemes in Figure 3.1.

The context performance model in Figure 3.1 illustrates how the different traffic classification schemes perform in relation to each other. The ends of axis present the maximum of that factor in a certain network (simulation) environment. Since we aim to observe the relative performance between the classifier schemes, the flow birthrate and flow space factors together with the application factor are normalized to the performance in the reference point where all flows are classified. The packet classification factor shows the percentage of all packets present in the measurement period that are classified to a particular service class.

The restrictions on transmission rate, i.e., the methods with which bandwidth should be allocated to connections are definitely of interest, but deserve a separate committed research effort. Connection setup delay is a protocol and implementation dependent metric and varies according to the technological environment used. Therefore, the other restrictions for real life traffic classification: the bandwidth/transmission rate, and the flow setup delay mentioned in [7] are not included in the context performance model in this work.

3.5.3 Content evaluation

A measurement based policy rule creation scheme should be designed so that applying the subsequent policy increases the quality of service the user is experiencing. The user should see the policy rules that the traffic classification is creating as beneficiary,

being able to offer appropriate handling to different traffic flows. As we now have a simple model with which to evaluate the context performance of a policy rule list, we start developing a model that assesses the content performance. By this we mean the evaluation of the actual rule lists produced by traffic classification.

Content evaluation is the analysis of the applications that fall into the different classes and comparison of classification decisions in different traces. The evaluation of classification means observing the classes in terms of the measured properties: applications, flows and packets.

The content evaluation method we propose focuses on evaluating the classification results in relation to previous results and comparing simple class statistics. We will compare the statistics to the data produced when all flows are classified. Then we will observe the statistics to establish the similarity of the class behavior in different traffic traces. Finally we will observe the statistics to spot differences produced by different classifiers with the same trace to see the characteristics of different classifier types. This will show us how the behavior of applications changes between traces and from class to class. The evaluation method consists of two components:

1. The first method observes the statistics of a traffic class. The statistics gathered are the mean, variance, minimum and maximum of the packet and flow count produced by applications classified to a class. Furthermore, since the mean and the variance obviously will change from one trace to another, we will also calculate the coefficient of variation $\frac{\sqrt{variance}}{mean}$ [55] that indicates the relative variation in packet and flow count data in a traffic class. The observations on the value of the coefficient of variation focus on observing the differences of the value between different traffic classes. The differences or similarities between traffic classes are then taken as indication on how different or similar the traffic class contents are. Some conclusions on class behavior might also be made based on the absolute value of the coefficient of variation, however, this is not the primary intent.[1]

2. The second method subjectively evaluates the lists of applications that result from the classification. The aim with this method is to assess the relevance of the classified applications based on a priori knowledge on application type and behavior. This method is the obvious one, but its results are subjective and dependent on the evaluator.

We will show the measurement and classification results in packet-flow –space as shown in Figure 3.2. We plot the applications by the application identifier a' (the TCP/UDP source port number *Sport* and the protocol identifier *Proto*) into the packet-flow –space using eqs. 3.9 and 3.10. We will also show the *mean* of eqs. 3.9 and 3.10 per class. Based on the standard deviation calculated for eqs. 3.9 and 3.10 in a class we will show a rough estimate on the area where the typical applications lie in the packet-flow –space by plotting a line of length $\frac{stdev}{2}$ in packet and flow dimensions to indicate the statistical range of applications in the class.

[1]The smaller the coefficient of variation is the more the applications in a traffic class behave in a similar manner. Larger coefficient of variation in a traffic class indicates applications that behave differently from each other.

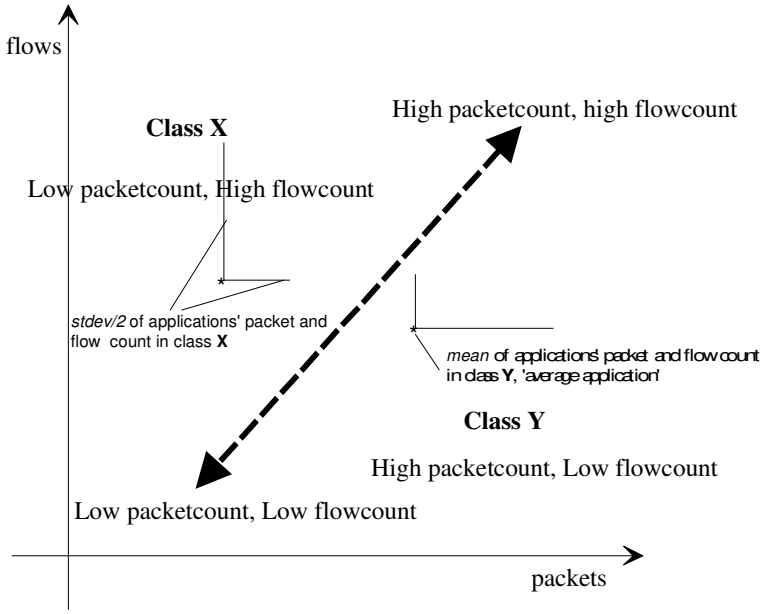


Figure 3.2: 2-dimensional packet-flow per application –space

3.6 Summary

This chapter has defined the concepts of packet, flow and traffic classification. The packet classifier reacts and performs on every packet that is detected in the system. The flow classifier, if implemented, needs to keep up with current and upcoming flows and it performs with the same intensity as the packet classification. Flow classification may also initiate the process of assigning resources to a flow. Traffic classification, on the other hand, observes the measured traffic characteristics it gets from packet and flow classification processes and, on relatively long intervals, it performs the traffic analysis that returns the rules of packet and flow classification as needed.

In this chapter we also developed a methodology to assess the traffic classification performance by means of context and content analysis. Context analysis is performed by observing simple and general factors in a simplified single IP router environment. These factors give an indication of the performance requirements of the IP router in an environment where traffic is classified.As far as classification is concerned being able to pinpoint the applications that produce either the majority or minority of the packets and flows and that behave in a particular manner might help to optimize the network performance and provide the possibility to offer users different types of service levels.

Finally, we developed a method of content analysis where we observe different packet and flow statistics of individual applications in a traffic trace. We also observe these properties in two-dimensional packet-flow –space to provide us some information on the application nature and characteristics. To further understand and aid in the analysis of the application lists we also map the classified applications to the 2-dimensional packet-flow –space. The data points in the packet-flow –space are determined by normalizing the measurement results for packet and flow count for each application a seen in the trace according to eqs. 3.9 and 3.10. The visualization of the normalized application statistics is done to enable comparisons in the packet-flow –space between different network envi-

ronments and to determine visually whether there exist areas in the packet-flow –space where different types of applications lie.

Chapter 4

Some methods for Internet traffic classification and notes on performance

4.1 Introduction

Most of the methods for flow classification have been presented and analyzed in the traffic-driven IP switching context [56, 29, 30, 14, 9, 32, 54]. Some of these classification methods take into account only the status and resource restrictions of the network equipment (routers), others rely on static lists of application information. The assessments made on these methods, however, do not place any weight on the effect the user is experiencing and it is debatable that the user is experiencing any significant performance improvement although the network equipment is functioning in an optimal fashion. Therefore, it may be argued that the policies implemented with these classification methods emphasize the overall resource optimization on the expense of individual user requests. The goal should be, nevertheless, to optimize the use of network resources but with more emphasis on the user's perceived quality of service [57].

This chapter focuses on the previously introduced methods for traffic and flow classification and their performance. The main purpose is to analyze the overall behavior of the detected application sets and use of classification resources as defined in Chapter 3. The results include observations on the size and contents of application lists that are produced when using particular classifiers. To achieve clarity we will, for the most part in this work, present the application lists as lists of TCP or UDP source port numbers that lie in the well-known port number area from 0 to 1023. For the upper regions of TCP/UDP port numbers we rely on the coherent behavior from the traffic classifier; if the traffic classification method picks up traffic flows of certain type in the well-known application (port) area, it is assumed that it can also pick up traffic flows of similar characteristics in the upper region of port numbers. In some cases, if the lists are short, we will show the complete application lists.

4.2 Test environment for the simulations and traffic traces used in this work

Throughout this work, we will be observing the performance of various traffic and flow classification schemes in a static simulation environment to enable the comparison of results from different traffic classification schemes. For the simulations, we use a set of six different one-hour long traffic traces obtained on two separate locations:

1. Three traffic measurements were made at Helsinki University of Technology in 1997 on a bridged 10 Mbit/s Ethernet local area network during (9 a.m. - 2 p.m.) office hours. The network served as the backbone network for the Department of Electrical and Communications Engineering serving some hundreds of users. Measurements were made with TCPDUMP.

2. The other set of measurements was obtained from the Internet Traffic Archive, where three traces from the Digital's Internet access point were used.

The simulation environment is fed with knowledge of the timestamp, protocol, the source and destination addresses, and the source and destination ports. As the analysis is performed, all sensitive information, including the packet lengths, were removed from the traces. The simulation environment is a connection-oriented environment. Connections, or flow states, are set up for those flows to which the rules provided by the traffic classification scheme indicate prioritization.

The traffic trace details are presented in Table 4.1.

Table 4.1: One hour traffic traces used in this work

Trace information			
Name	Location	Nr of packets	Media
ebb900	Helsinki University of Technology (HUT)/Campus Area Network 09:00 May 29, 1997	1107188	Ethernet
ebb115	HUT/CAN 11:50 June 6th, 1997	1007398	Ethernet
ebb130	HUT/CAN 13:30 May 27, 1997	1233970	Ethernet
dec-pkt-1 (dec1)[a]	Digital's primary Internet access point (DIAP) 22:00, March 8th, 1995	2983217	Ethernet
dec-pkt-2 (dec2)	DIAP 02:00, Thu March 9th, 1995	3467733	Ethernet
dec-pkt-3 (dec3)	DIAP 10:00, Thu March 9th, 1995	4086848	Ethernet

[a]All dec-pkt-x traces are freely available at http://ita.ee.lbl.gov/html/contrib/DEC-PKT.html

In the basic flow analysis of the network traffic, the granularity of the flow is defined to include IP source and destination address, protocol information and TCP or UDP

source and destination port fivetuples according to the packet parameters defined earlier. The connections, or flow state setups, are initiated by the first packet in a traffic flow and subsequent packets are counted in the connection. The flow state is held up for 60 seconds [30, 41] after the final packet in the flow has been seen in the flow. This characteristic is likely to increase the requirements for flow state space, or maximum flow state space, but it is also probable that more packets may be mapped to a connection. The same kind of environment has been suggested in [41] and used in various research activities before [29, 58, 30].

Relevant simulation parameter values are shown in Table 4.2.

Table 4.2: Simulation parameters for traffic and flow classification

Simulation environment parameters	
Parameter	Value
Flow timeout τ [a]	60 seconds
Flow granularity m [b]	IP address and TCP or UDP[c] port level (5-tuple) granularity, $m = (32, 32, 8, 16, 16)$.

[a] The time limit for consecutive packets in a flow. If exceeded, the flow is deleted and a new one has to be created.

[b] Granularity determines the level at which flow candidates are introduced.

[c] Difference between TCP and UDP is determined by *Proto*.

4.3 The reference point for performance study

For reference purposes, we introduce the "null" method of traffic classification, where all flows are classified as the first packet on the flow is observed. The first packet of the flow is also considered to belong to the flow as is also indicated in Table 3.3. The performance observed will set the reference point to which all other classifiers will be compared [56, 14, 32].

Since all flows are classified, the behavior at the reference point could be considered to be the worst there can be, given a particular flow definition and, therefore, all other classifiers perform not worse or better.

The pseudo code implementation for studying the performance at the reference point is shown in Table 4.3. The pseudo code in Table 4.3 shows that the traffic classification functionality is very simple: All existing application identifiers are considered to form the classification criteria, thus leading to a situation where all flows detected are prioritized to class 1 meaning the priority class.

Table 4.3: All flow classification

process Traffic_Classification

$$\mathbb{S}_{all} = \{(a, 1) \mid \exists p \in a\};$$

end Traffic_Classification;

4.3.1 Performance at the reference point

The context analysis performance results at the reference point where all flows are classified are shown in Table 4.4. After the flow timeout value has passed without any packets observed, the flow state is deleted. Similar analysis has been performed in [56, 14].

Table 4.4: Context analysis performance results at the reference point

Basic simulation results at the reference point				
Network[a]	Flow birthrate factor, B_{base} at time t_F [b]	Flow space, S_{base} [c]	Classification factor, C_{base} [d]	Amount of applications, A_{base}
dec1	2830 at 3568s	4579	100%	7837
dec2	2848 at 3574s	4255	100%	7483
dec3	5464 at 3596s	8576	100%	12141
ebb900	1757 at 3557s	2611	100%	9211
ebb115	1470 at 3642s	2776	100%	8614
ebb130	1296 at 3537s	2309	100%	7682

[a] Refer to table 4.1
[b] referring to the maximum of flow setups in one second in the trace
[c] Maximum number of simultaneous connections during the trace period.
[d] % of packets classified.

Results in Table 4.4 show that the flow state space factor depends on the particular network characteristics, such as the network size or geographical location. In the dec-traces the time of day seems to be also affecting the flow factor. The use of flow state space is quite moderate in absolute terms but varies somewhat from trace to trace. The number of applications detected in different networks is relatively constant. However, dec3 shows a substantial increase in the application count. In conclusion, this type of traffic classification is not meant to achieve any real differentiation of traffic but offers a performance comparison point.

Next, we observe the statistics on packet and flow shares per application in Table 4.5. Table 4.5 shows that the coefficient of variation tends to be larger for flow data in dec-traces, whereas in ebb-traces the coefficient of variation is larger for packet data. Therefore, it may be concluded that the dec–traces contain applications that vary more in temporal behavior than the applications in ebb-traces. This statement is further backed up by looking at the maximum values of the flow and packet shares in the traces. In ebb– and dec–traces the maximums of the packet shares are practically equal whereas the maximums of flow shares vary significantly between the trace-environments being higher in the dec-traces.

Looking at the coefficient of variation in all of the trace environments we see that the values range between 16 to 51, the combined average for both values being approximately 27. Due to the seemingly large values it seems that all of the traces contain applications that behave quite differently from each other.

In Figures 4.1 to 4.4 the locations of applications are shown in the 2-dimensional packet-flow –space. We also show the 'average application' based on the mean of the packet

Table 4.5: Content statistics for all traces with all flows classified

Content statistics for dec- and ebb-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
dec1	all packets	0,0128%	0,2805%	0,00003%	18,6375%	22
dec1	all flows	0,0128%	0,6547%	0,0006%	55,5973%	51
dec2	all packets	0,0134%	0,2468%	0,00003%	12,2226%	18
dec2	all flows	0,0134%	0,584%	0,0006%	47,5911%	44
dec3	all packets	0,0082%	0,2017%	0,00002%	16,0185%	24
dec3	all flows	0,0082%	0,4003%	0,0003%	42,9367%	49
ebb900	all packets	0,0109%	0,2463%	0,00009%	18,1552%	23
ebb900	all flows	0,0109%	0,1826%	0,0014%	11,1828%	16
ebb115	all packets	0,0116%	0,2329%	0,0001%	13,5456%	20
ebb115	all flows	0,0116%	0,1892%	0,0015%	11,5094%	16
ebb130	all packets	0,013%	0,2819%	0,00008%	16,4349%	22
ebb130	all flows	0,013%	0,2218%	0,0019%	12,9275%	17

and flow shares as indicated in Table 4.5 and based on variance data we show the range in packet and flow dimensions that outlines the range of the average application.

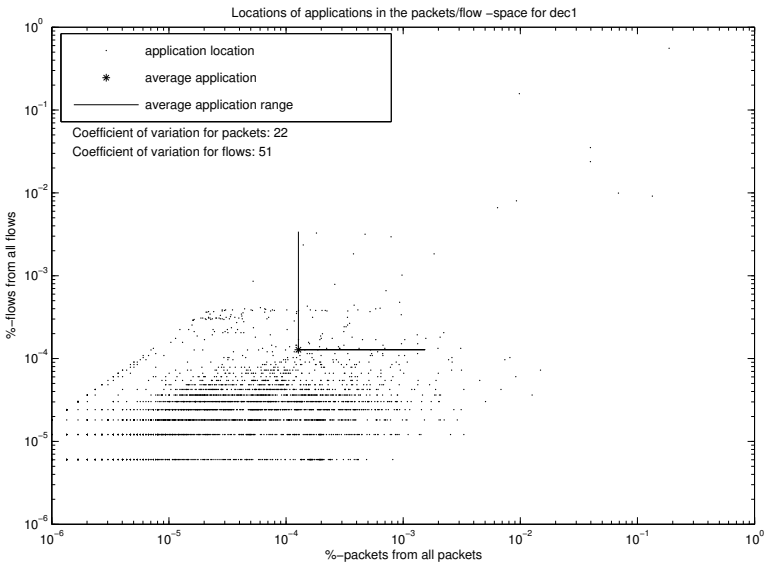
Figures 4.1 to 4.4 support the findings in Table 4.5 that the variation in flow shares (y-axis) of the ebb-traces is lower than the variation in dec-traces. Furthermore, the applications in ebb-traces seem to vary more in packet shares than in flow shares. This indicates that the applications in ebb-traces behave smoothly and have varying presence.

At this point we extend the use of the characterizing measurements by observing individual applications measurement results and changes therein from trace to trace. This type of analysis tries to characterize the changes in the measurements of the same application within the same network. The less changes there are in the measurement results the more likely the application is classified similarly. In this work we will show the changes that an application experiences by observing the movement in the packet-flow –space from trace to trace in the same network environment. Specifically, the movement is defined by calculating for an application a the normalized distance $r_{a,norm}$ for (C_{norm}, F_{norm}) in traces $\mathbb{T}$ and $\mathbb{T}_{ref}$. The distances are normalized against the maximum distance that an application can move in the packet-flow –space:

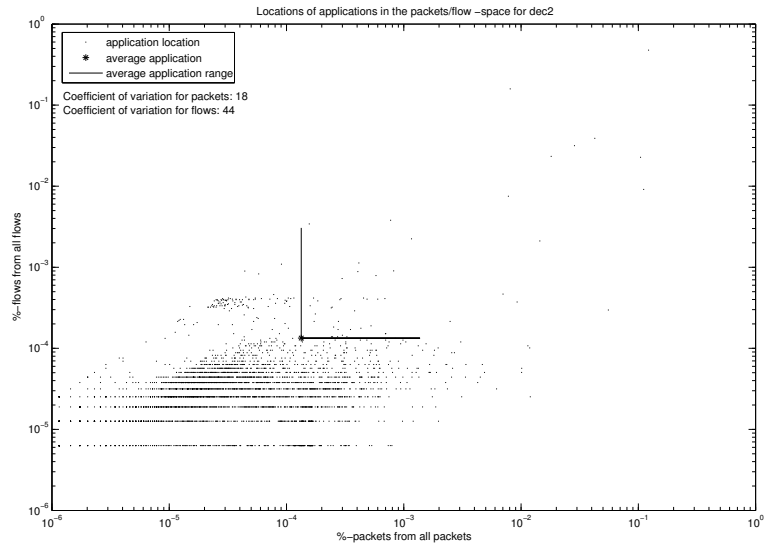
$$r_{a,norm} = \frac{|(C_{norm}(a, \mathbb{T}_{ref}), F_{norm}(a, \mathbb{T}_{ref})), (C_{norm}(a, \mathbb{T}), F_{norm}(a, \mathbb{T}))|}{\sqrt{2}} \tag{4.1}$$

Next we calculate the distances that the applications travel in the packet-flow –space from a reference trace in a network environment to another trace according to Eq. 4.1. The distance data is determined only for those applications that exist in both of the traces. The distances are sorted to descending order starting from the largest (normalized) distances and shown in Figures 4.5 and 4.6. Some familiar applications are indicated in the figures. Some basic statistics of the movement in dec- and ebb-environments are shown in Table 4.6.

We can see from Figures 4.5, 4.6 and Table 4.6 that the movement of applications from one trace to another is generally quite small and that the majority of the applications tend to behave similarly in regards to packet and flow shares. However, some of the



(a) dec1



(b) dec2

Figure 4.1: Application locations in logarithmic packet-flow –space /dec1 and dec2.

more interesting applications (indicated in the figures) such as 53/udp, 80/tcp, 22/tcp and 23/tcp to mention a few, seem to be moving quite much up to tens of percent of the maximum movement. To be able to classify these applications based on the packet-flow -information to proper classes requires the classifier to be dynamic in placing the classification borders in the packet-flow –space. Furthermore, the share of applications that are found in both the reference trace $\mathbb{T}_{ref}$ and in the $\mathbb{T}$ is at its highest only 50%. Although the number of traces that are compared is quite low, we claim that a method that selects applications in a dynamic fashion should exist.

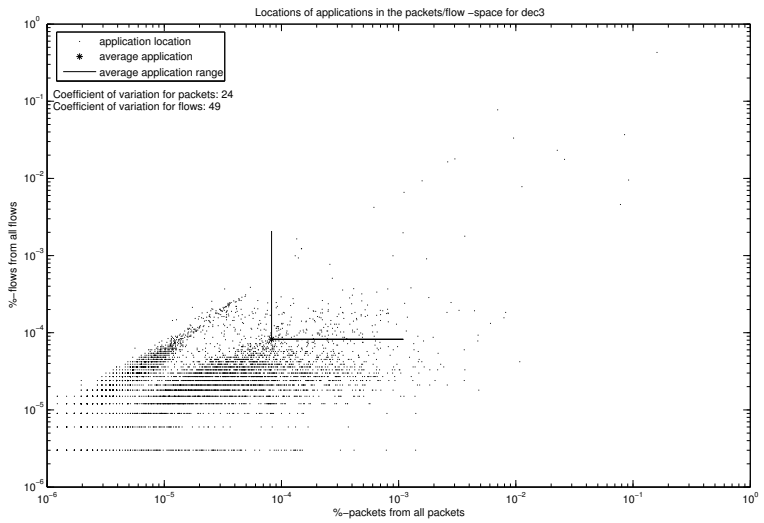


Figure 4.2: Application locations in logarithmic packet-flow –space /dec3.

Table 4.6: Statistics of the distance data

Statistics of the normalized distance data					
Traces	App. count[a]	Mean	Variance	Min	Max
dec1-dec2	50%	$3,5 \cdot 10^{-4}$	$4,7 \cdot 10^{-5}$	$2,6 \cdot 10^{-7}$	$1,1 \cdot 10^{-1}$
dec1-dec3	30%	$2,9 \cdot 10^{-4}$	$3,7 \cdot 10^{-5}$	$6,5 \cdot 10^{-7}$	$4,1 \cdot 10^{-1}$
ebb900-ebb115	37%	$3,1 \cdot 10^{-4}$	$1,2 \cdot 10^{-5}$	$8,5 \cdot 10^{-7}$	$9,5 \cdot 10^{-2}$
ebb900-ebb130	47%	$3,3 \cdot 10^{-4}$	$1,5 \cdot 10^{-5}$	$1,7 \cdot 10^{-7}$	$1,2 \cdot 10^{-1}$

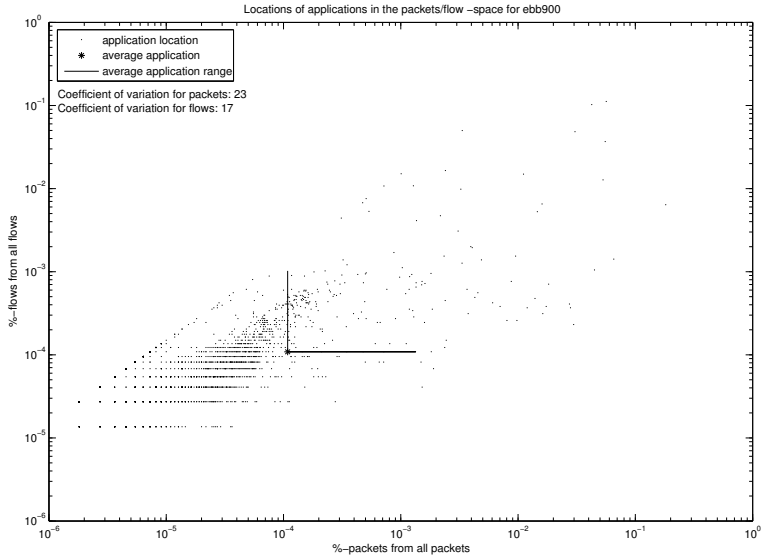
[a]Share of applications found in both traces compared to the total amount of applications in the first trace.

4.4 Methods of traffic classification

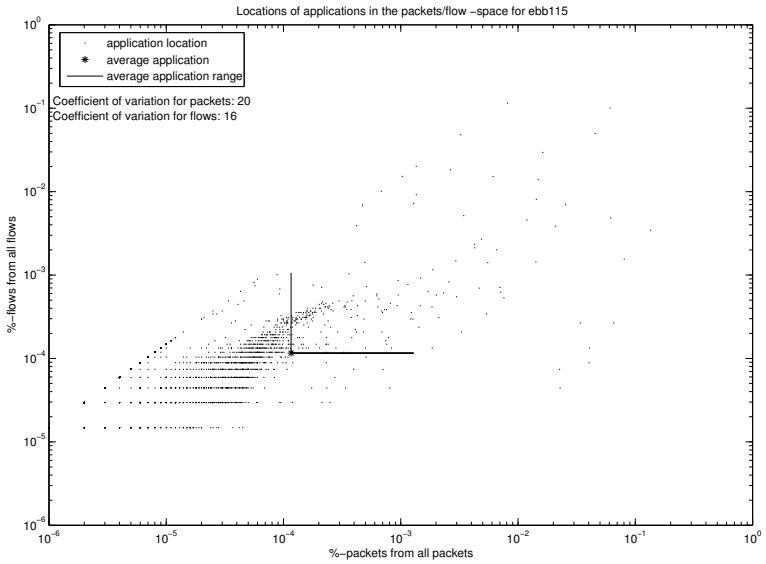
4.4.1 Introduction

The traffic classification methods provide sets of rules to determine the criteria with which traffic is classified or divided into traffic classes. The classification methods may or may not use feedback from the network or the user. Some suggestions on what should be observed and what kind of rules should be determined include [59] (see also Section 3.1 for a discussion on what there is to measure in a network):

- The method that observes the number of packets on a flow candidate and establishes the flow state after a certain threshold is reached is called packet count flow classification. It has been widely studied in [56, 29, 30, 14, 32, 54, 60, 6]. Furthermore, the dynamic version of the packet count classifier, where the packet count threshold is dynamically updated depending on the use of router and classification resources, is studied in [31, 61, 62, 63, 64].



(a) ebb900



(b) ebb115

Figure 4.3: Application locations in logarithmic packet-flow –space /ebb900 and ebb115.

- Measuring the packets based on their (payload) size and then determining the packet size values as classification bounds. The packet size distributions have been observed in [65, 8] but no significant characteristics could be concluded. Some work was also done in [66] to visually distinguish the application signature of the www-application by observing packet sizes and packet size distributions.

- Measuring the distribution of the packet arrival process of a flow and then determining the shape and form of these packet arrival distributions. This method of

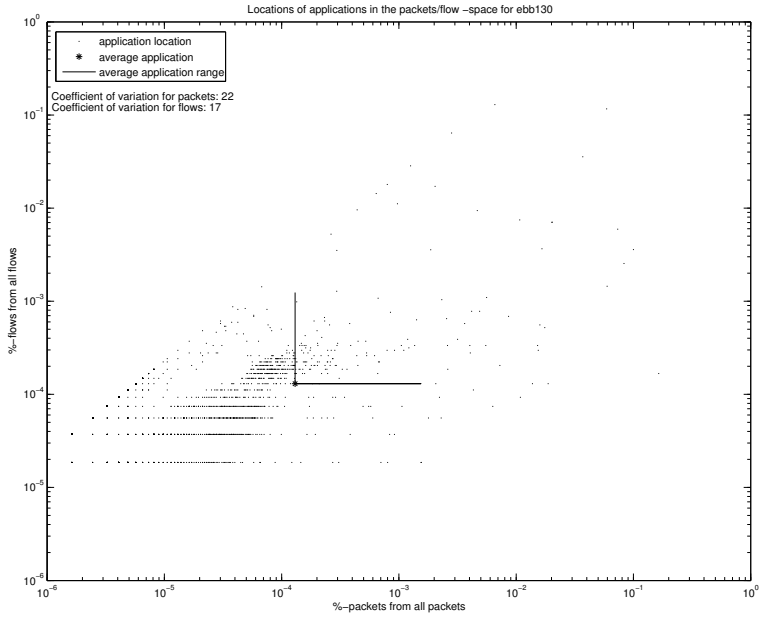


Figure 4.4: Application locations in logarithmic packet-flow –space /ebb130.

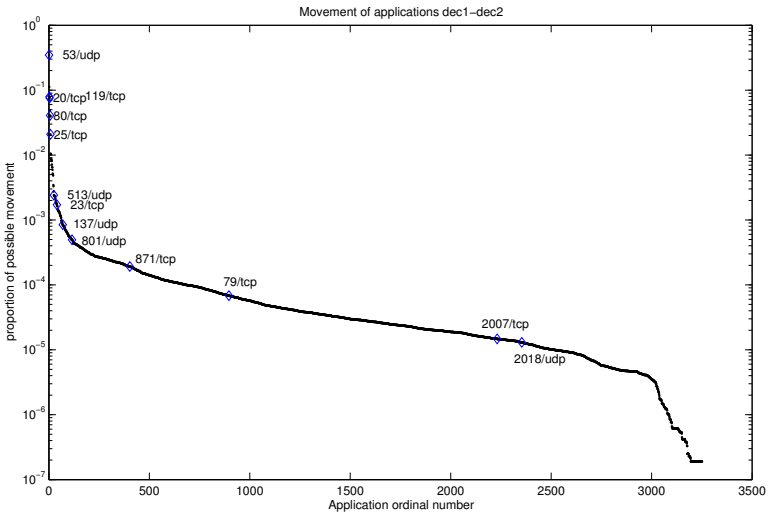
classification is suggested in [33, 34] and the feasibility is superficially analyzed in [65, 8].

- Classifying packets based on the content in their headers, e.g., port numbers; and then determining the content values for classification. The static form of this classifier appeared first in [56, 29] and has been further studied in [9]. The work in this thesis, in the coming chapters, introduces the method of dynamically choosing the application identifiers based on aggregated packet-flow –measurements.

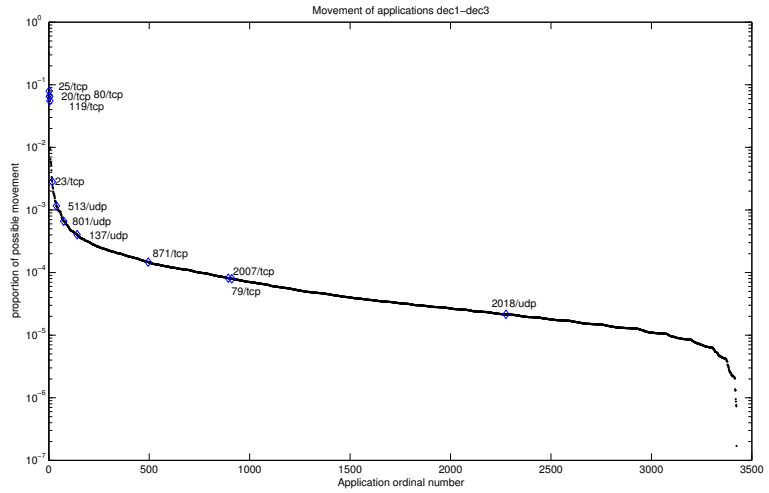
4.4.2 Packet count classifier

A packet count classifier is a classifier that makes a prediction on the future behavior of the flow based on the past information it has gathered. The traffic classification process determines how many packets are needed before a flow is set up. The aim of this approach is to reduce the workload of the forwarding processor by assigning future packets under a flow identifier. The traffic classification process behind the packet count classifier could also make suggestions on the timeout values.

The packet count classifier requires that X packets arrive before establishing an active flow. Before X packets have arrived the flow is called a candidate flow [67, 6]. As a packet arrives into a network node, an entry in the flow candidate table is created, or updated, provided the packet can not be assigned to an active flow. If the packet count threshold is reached, the candidate flow is deleted, an active flow is created, and the flow is provisioned appropriately. This classification depends on the application; in IP switching [32] and in the ATM MPOA [68, 69] the flow would be directed to its own connection, in a connectionless environment the flow could be assigned a header which



(a) dec1-dec2



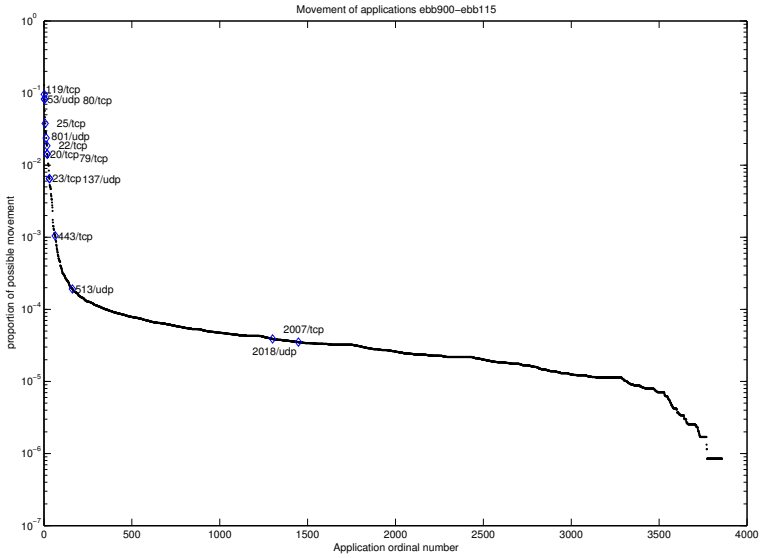
(b) dec1-dec3

Figure 4.5: Application movement between dec-traces

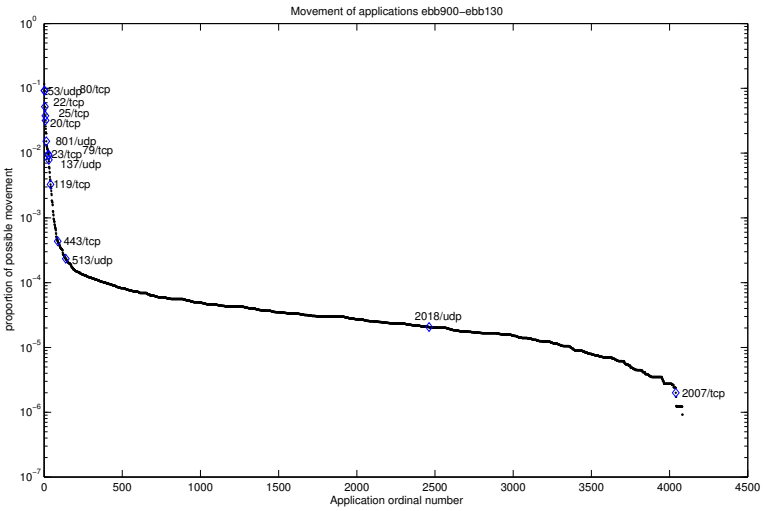
implied a better priority handling in the network [50, 51, 52].

The use of the packet count classifier as a method for resource based traffic classification has been studied in [29, 30, 14, 9, 32, 70, 54, 44, 60, 6]. These studies have mainly concentrated on the reduction of flow setup workload when implementing fine granularity per-flow based IP switching with minor emphasis on the active flow statistics. Particularly, with the packet count classifier the flow setup delay easily grows to intolerable values increasing the possibility of delivering packets out of order [30].

If we consider the practical use of the packet count classifier, the natural choice would be to apply the classifier in a per-flow environment. This means that if we are to offer QoS, or CoS, to the flow we need to convey related state information throughout the network. While possibly feasible in the edges, or in restricted areas, of the network this



(a) ebb900-ebb115



(b) ebb900-ebb130

Figure 4.6: Application movement between ebb-traces

scheme may not be efficiently implemented through the core network on the account of the scalability issues [7].

The implementation uses the flow candidate timeout value as a mechanism to remove outdated candidate entries that have small probability of reaching the packet count value, X. There is no emphasis on the packet header contents. Some of the implementation details and obstacles concerning the candidate table in particular, have been reported in our previous work [6].

4.4.3 Dynamic packet count classifier

The packet count classifier may also be used to balance and control the use of IP router resources in the appropriate environment if the threshold value is altered as has been suggested with the adaptive packet count [31, 54, 61, 62, 63, 64] approach. The control of resource usage is even more important if the actual implementation of the network functionality is a complex issue, e.g. as in MPOA [69, 62].

An example of a dynamic resource based traffic classification is the adaptive packet count [54, 61, 63, 64] approach. The use of IP router resources is measured by the traffic classification process and the adaptation of the packet count threshold is based on the measured status of the IP router system. The goal is to minimize the utilization difference of the resources. In the scheme, the traffic classification process is on the tight feedback loop from the network equipment resource status to the parameters of the flow classifier.

4.4.4 Inter-arrival time classifier

Traffic classification process may also be based on measuring the inter-arrival times of the packet streams [33]. Traffic flows supposedly have realtime properties if the distribution of the arrival process is unimodal and non-realtime properties if the distribution is bimodal [34, 65, 8]. This assumption is somewhat validated in [65, 8]. The monitoring of the inter-arrival distribution could be limited to certain traffic flows based on the TCP/UDP port numbers [33, 34]. The work done in [33, 34] is, however, void of any effort on picking these port numbers dynamically from the network traffic. Therefore, the inter-arrival time -classifier would perform like the static application classifier (introduced next) with an additional application filter in the form of monitoring the inter-arrival distribution.

4.4.5 Static application classifier

The user based classification methods aim to classify flows produced by applications that are considered important to the user because of their interactive nature or some other user oriented property. The most common form of user based traffic classification is to use application lists that are usually predetermined by a network manager. This type of a classifier first appeared in [56, 29] where the applications were identified with TCP port numbers. The application list was static and consisted of a fixed amount of "well-known" applications. It has been since used as a part of an analysis of the traffic driven label mapping in MPLS [71].

The pseudo code implementation of the static application classifier is shown in Table 4.7. The static nature of the traffic classification process is indicated by making the rule set $\mathbb{S}$ constant. The changes in the application list require the intervention of the network manager. In large networks with varying application usage, it would be hard to react to traffic and application profile changes using the static application classifier. To determine the list of prioritized applications we could measure the network and make an intelligent choice based on some measured property or properties. Depending on the network environment and the total number of applications present in the network, however, it might be difficult to detect all the applications eligible for prioritization.

Table 4.7: Pseudo code for static application classifier

process Traffic_Classification
const $\mathbb{S}_{SAC} \subseteq \mathbb{S}_{all}$;
end Traffic_Classification;

4.4.5.1 Context performance analysis

With the static application classification, we pick out a set of important applications by hand from the network. The importance is difficult to define exactly but refers generally to interactive applications. Interactive, in this context, is also loosely defined to be any application that includes a human user on the other end of the packet stream interacting with another human user or with a computer.

The static application classifier uses a set of applications that has been determined by observing the application sets when all flows were classified and when using packet count classifiers. We will also take into account the considerations, mentioned in previous studies [29, 40], for priority applications. We aim to combine a compact and limited application set from all of the traces used. This process should resemble the actual process of choosing an application set by a human network manager. We note that by choosing a different set of applications the performance could be significantly changed. The final application set of fourteen (14) applications is shown in Table 4.8.

Table 4.8: The selected application set, $\mathbb{S}_{SAC}$, for the Static Application Classifier

Application set $\mathbb{S}_{SAC}$	
Port number and protocol($Sport/Proto$)	Application description
20/tcp	ftp data connection
22/tcp	Secure Shell
23/tcp	Telnet
25/tcp	Simple Mail Transfer Protocol, SMTP
69/tcp	Trivial File transfer
70/tcp	Gopher -service
80/tcp	WWW-protocol, HTTP
110/tcp	Protocol used for mail-sessions, POP
119/tcp	Network News Protocol, NNTP
443/tcp	Secure http
513/tcp	login service
514/tcp	shell service
995/tcp	secure pop3

These applications have a definite and direct interest of the user thus giving ground for prioritization. We limit the applications to those that are particularly well known; applications that have a Well–Known TCP port number[1] ranging from 0 to 1023.

[1] http://www.iana.org/assignments/port-numbers

The simulation results of the static application classifier are presented in Table 4.9 and in Figure 4.7.

Table 4.9: Context performance of the static application classifier

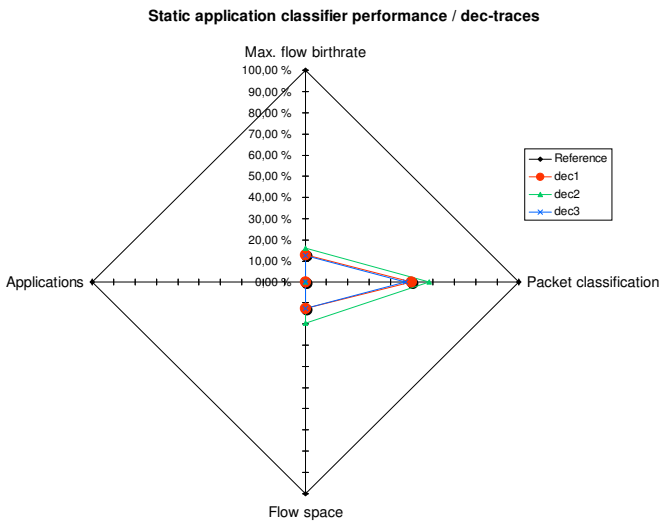
Basic simulation results for the static application classifier				
Network[a]	Flow birthrate factor, B_{SAC}	Flow space factor, S_{SAC}	Packet classification factor, C_{SAC}	Application factor, A_{SAC}
dec1	12,83% at 3518s	12,56%	49,76%	0,14%[b] / 8 of 14 applications[c]
dec2	16,12% at 3538s	19,61%	57,85%	0,15%/ 8 of 14 applications
dec3	12,54% at 3578s	12,53%	46,77%	0,14%/ 9 of 14 applications
ebb900	37,34% at 3582s	49,72%	17,89%	0,18%/ 11 of 14 applications
ebb115	32,86% at 3621s	39,27%	51,40%	0,18%/ 11 of 14 applications
ebb130	50,31% at 3527s	38,78%	51,12%	0,25%/11 of 14 applications

[a]Refer to table 4.1
[b]Of all applications found in $\mathbb{P}_T$
[c]Of the static set of fourteen applications as described in Table 4.8

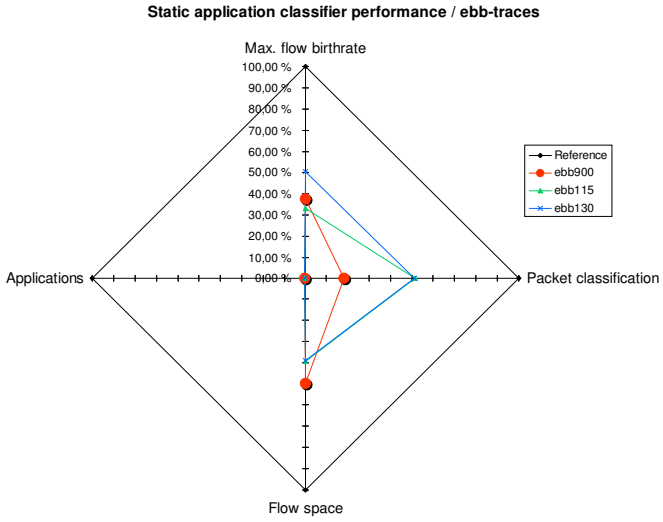
Table 4.9 and Figure 4.7 show a significant reduction of at least 50% of flow space usage and reduced flow birthrate factor for the static application classifier compared with the reference point. The packet classification factor remains relatively constant around 50% indicating that the use of the selected 14 applications in these networks produces an equal portion of the total network traffic. In the ebb900 -trace, based on the higher use of flow space and decrease in the packet classification factor compared to other ebb-traces, the application set seems to be out of place indicating either the difficulty of choosing an adequate and descriptive set of applications to describe the network profile or a significant change in the network application profile. Looking from the context performance point of view the static application classifier may produce slightly unpredictable behavior when network environment is changed.

4.4.5.2 Content analysis

The content statistics of the dec- and ebb-traces for the static application classifier are shown in Tables 4.10 and 4.11. Looking at the content statistics of both the dec– and ebb–traces in Tables 4.10 and 4.11 we can see that the means and the standard deviations of the packet and flow data are roughly the same in class 1 and class 0 respectively in different traces. Furthermore, Class 0 (the default class) coefficients of variation display in all traces values that are equal to or even higher than the values observed in the reference point. Class 1 (the priority class) has roughly equal values of the coefficient of variation in all of the traces and Class 1 values are an order of magnitude lower than



(a) dec-traces



(b) ebb-traces

Figure 4.7: Performance of the static application classifier

those of Class 0. Partly this phenomenon is explained by the low number of classified applications. However, looking at Table 4.8 we can see some applications that are related to each other via the type of usage: for instance, the telnet–application (port 23) and ssh–application (port 22) have partly a similar purpose of use in the network. We conclude that, based on these figures, the static application classifier has succeeded in selecting applications to the traffic classes that behave in a similar fashion within the class. Were the applications selected in a random fashion the value of the statistics would be closer to those of the reference point with all flows classified. However, the small number of applications in Class 1 means that the confidence for the classification results is still low. Nevertheless, the static application classifier may be used with caution in other networks without significant changes to the application set $\mathbb{S}_{SAC}$.

The placing of some applications within the packet-flow –space in network traces is shown

Table 4.10: Content statistics for dec-traces with static classification

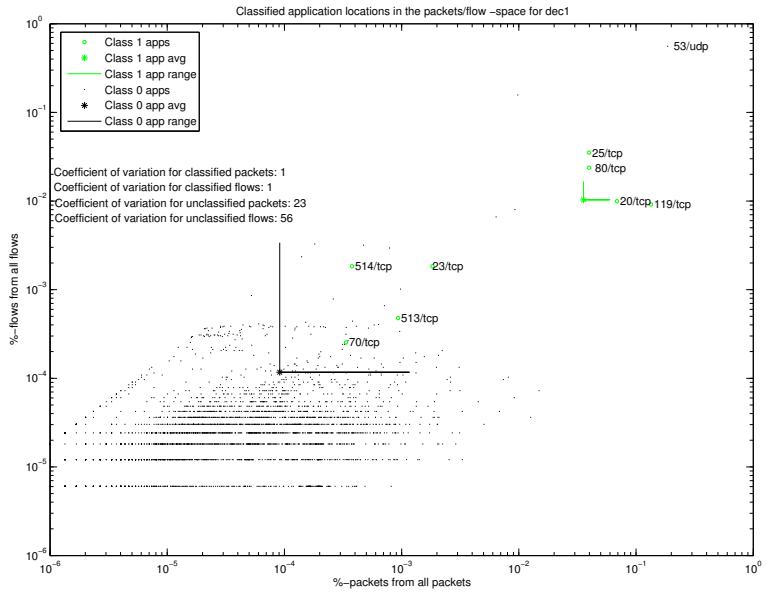
Content statistics for dec-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
dec1	Class 1 pkt	3,569%	0,1458%	0,034%	13,396%	1
dec1	Class 1 flw	1,030%	0,040%	0,0254%	3,516%	1
dec1	Class 0 pkt	0,009%	0,172%	0,00003%	18,637%	23
dec1	Class 0 flw	0,012%	0,5244%	0,0006%	55,597%	56
dec2	Class 1 pkt	3,657%	0,1452%	0,008%	11,112%	1
dec2	Class 1 flw	1,245%	0,047%	0,0360%	3,898%	1
dec2	Class 0 pkt	0,009%	0,129%	0,00003%	12,223%	17
dec2	Class 0 flw	0,012%	0,456%	0,0006%	47,591%	48
dec3	Class 1 pkt	2,128%	0,107%	0,002%	8,468%	2
dec3	Class 1 flw	0,854%	0,041%	0,0003%	3,687%	2
dec3	Class 0 pkt	0,007%	0,171%	0,00002%	16,019%	26
dec3	Class 0 flw	0,008%	0,398%	0,0003%	42,937%	52

Table 4.11: Content statistics for ebb-traces with static classification

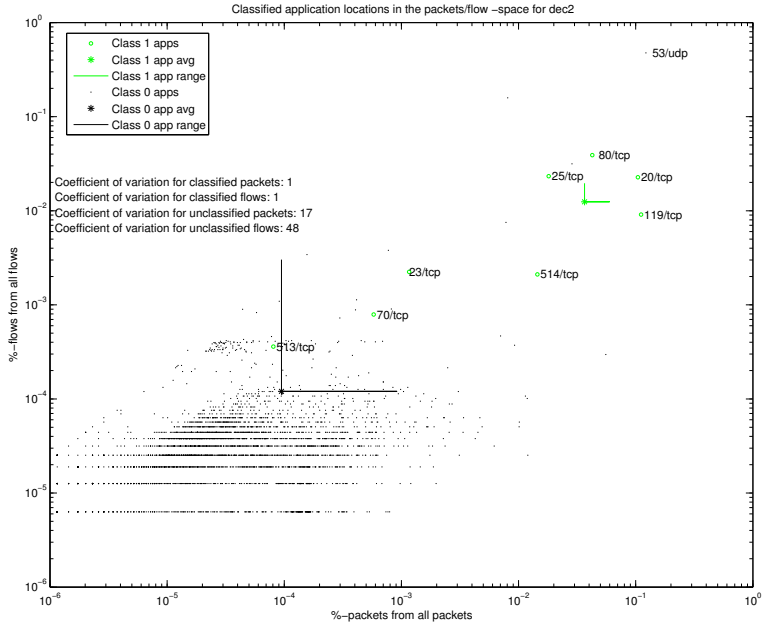
Content statistics for ebb-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
ebb900	Class 1 pkt	0,963%	0,064%	0,001%	5,646%	2
ebb900	Class 1 flw	1,3247%	0,12%	0,0027%	11,1828%	2
ebb900	Class 0 pkt	0,010%	0,238%	0,0001%	18,155%	24
ebb900	Class 0 flw	0,0093%	0,1398%	0,0014%	10,2406%	15
ebb115	Class 1 pkt	3,0602%	0,1739%	0,0001%	13,5456%	1
ebb115	Class 1 flw	1,2502%	0,1061%	0,0015%	10,0193%	2
ebb115	Class 0 pkt	0,0077%	0,143%	0,0001%	8,040%	19
ebb115	Class 0 flw	0,0100%	0,149%	0,0015%	11,5094%	15
ebb130	Class 1 pkt	1,8450%	0,118%	0,0006%	7,3444%	2
ebb130	Class 1 flw	1,1927%	0,122%	0,0019%	11,6482%	3
ebb130	Class 0 pkt	0,0102%	0,2510%	0,0001%	16,4349%	25
ebb130	Class 0 flw	0,0112%	0,1773%	0,0019%	12,9275%	16

in Figures 4.8 to 4.11. We note that the applications seem to exist in groups in the packet-flow –space, and these groups may be found in different network environments. We can see that applications like ssh and telnet in ports 22/tcp and 23/tcp respectively cluster near each other in the high packet count - low flow count area over different traces. The same thing seems to be occurring with the www-service (port 80/tcp). Note also, that especially in the ebb-traces the www-service and dns-service (53/udp) are quite near each other in the high packet count - high flow count area. However, some applications tend to be positioned only very generally to a certain area in the packet-flow –space. Based on this, we can assume that established applications tend to behave in the same way relative to other applications in the packet-flow –space over different network environments. This statement is also backed up in Figures 4.5, 4.6 and in Table 4.6 that show the relatively

small movements of common applications between traces.



(a) dec1



(b) dec2

Figure 4.8: Locations of selected applications for the static application classifier - dec1 and dec2

We also show the mean values of packet and flow shares of Class 1 and Class 0 in different traces together with the estimated range of the applications. We can see that Class 1 seems to have much more concentrated behavior in packet-flow –space. This conclusion

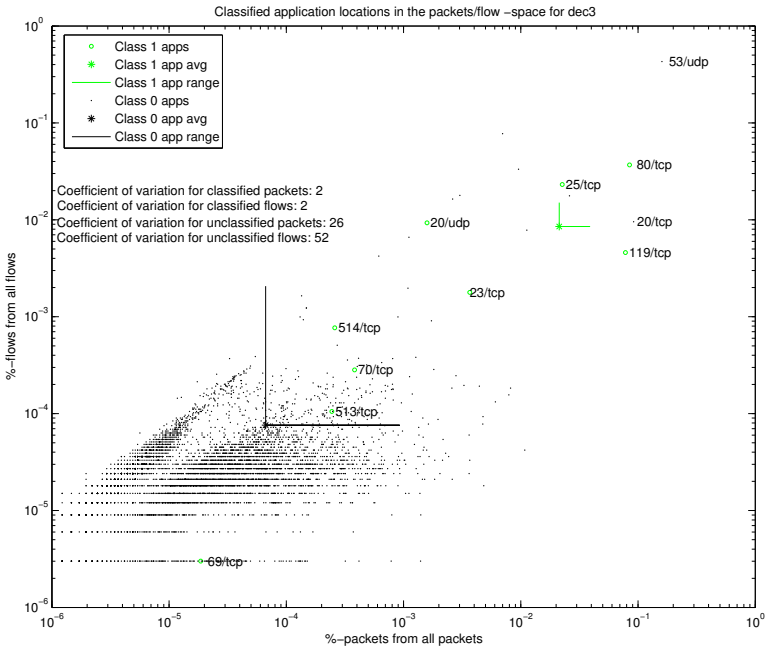


Figure 4.9: Locations of selected applications for the static application classifier - dec3

is in line with the results found in Tables 4.10 and 4.11.

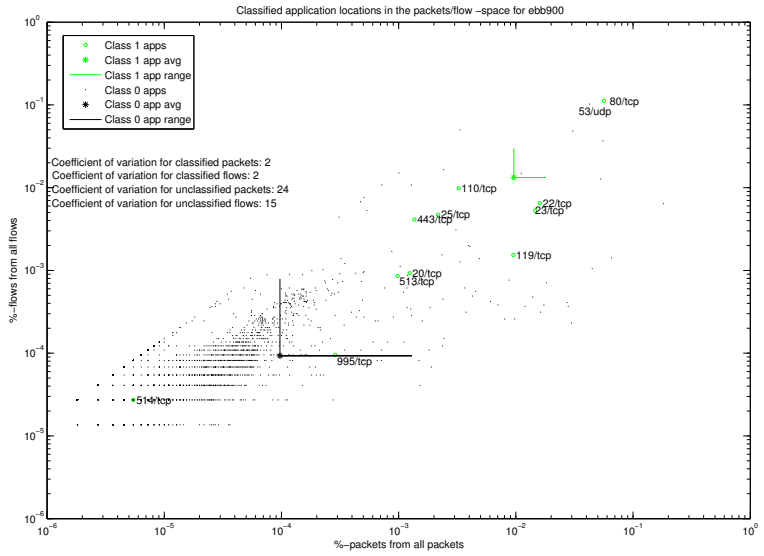
4.4.6 Measurement based threshold classifier

Approaches have been proposed that determine the application lists based on measured properties of the network traffic. The key idea in these approaches is to find traffic characteristics that may indicate application flows important to the user.

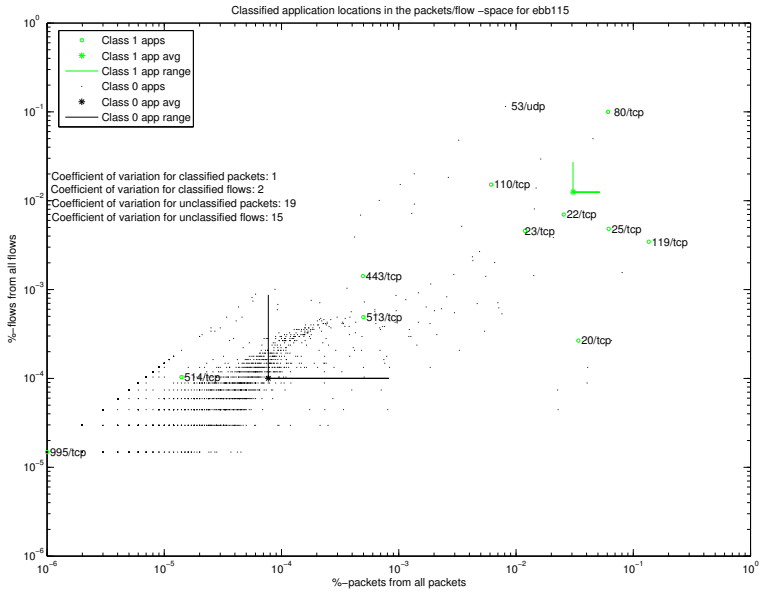
A suggestion to dynamically determine the applications for classification is given in [11] in the form of a threshold classifier. This classifier functions in the following manner:

1. It first measures in the network the aggregated packets/flows -ratio per application.
2. It then orders these measurement results to a list where the ratios descend from the largest to the lowest.
3. Then a classification threshold is placed so that a chosen percentage of the applications are classified to a priority class.

For instance, if the network was measured to contain one thousand applications, and the threshold would be placed to 20%, the list of classified applications would include every application up to the 200th. The placing of the threshold is arbitrary depending on the actual traffic classification process. The traffic classification process in [11] is void of any attempts to adjust the threshold and could be described static. This method classifies to



(a) ebb900



(b) ebb115

Figure 4.10: Locations of selected applications for the static application classifier - ebb900 and ebb115

high priority applications with a high packet count and a low flow count, thus favoring applications that send high number of packets but occur rarely in the network.

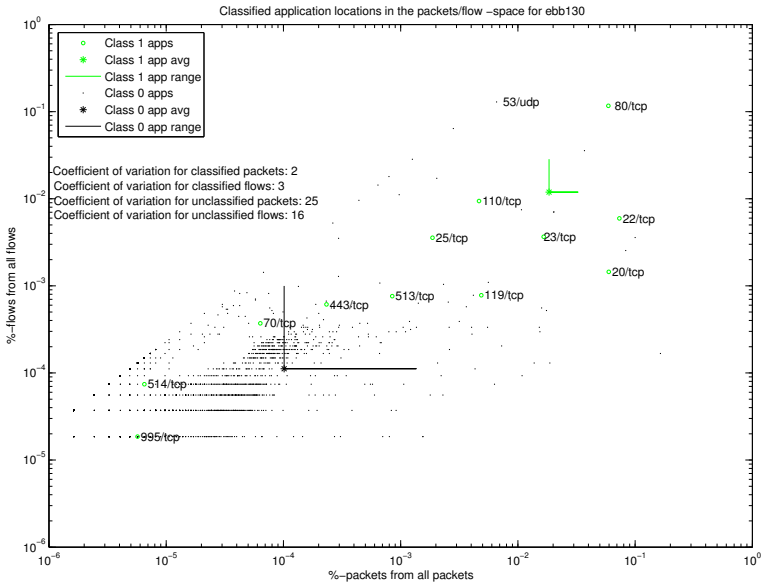


Figure 4.11: Locations of selected applications for the static application classifier-ebb130

4.4.6.1 Context performance analysis

The threshold classifier is a step to a dynamic direction from the static application classifier. The application set for high priority traffic is dynamically picked out based on an arbitrarily chosen percentage of the applications. Based on the packets/flow -ratio those applications with the highest ratio are put to the top of the list. In this work the applications are ordered by the packets/flows -ratio so that a threshold is set to 2% and 10% of the top applications to be classified to higher priority. The context performance of the threshold classifier is outlined in Table 4.12 and Figure 4.12.

We can see that the numbers for the flow birthrate factor and the use of flow space are somewhat varying but there is an observable reduction compared to the all flow classification at the reference point. Comparing the threshold classifier to the static application classifier the results are ambiguous, especially in regards to the flow birthrate factor. The use of flow space with the 10% -threshold is quite conservative, but rises occasionally (in ebb900 and ebb115) to a relatively high level. Most notably, the classification factor is high indicating that the applications with above threshold packet/flow –ratio seem to produce most of the packets. In ebb-traces, choosing just 10% of the top applications we can see over 80% of the packets classified to a higher priority. Even the lowest level of packets classified to high priority is still over 60%. Taking into account the high packet classification factor, the flow birthrate factor and the use of flow space are well contained. The majority of the packets are classified with less than 30% use of flow setup capacity (flow birthrate) and flow space. The only exception to this is found in ebb900-trace that uses quite a lot of flow setup and flow space resources. All in all, the threshold classifier seems to produce a predictable context performance in different network environments. However, the actual level of performance is dependent on the network.

The performance results presented in Table 4.12 for the threshold classifier contain an

Table 4.12: Performance results for the threshold classifier

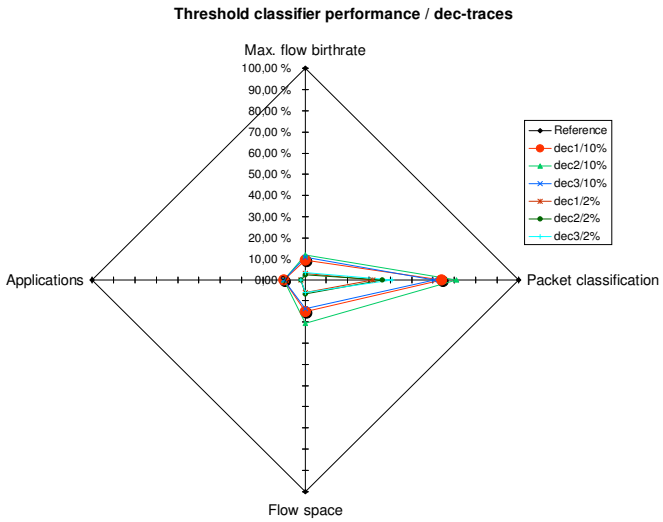
Basic simulation results for the threshold classifier				
Application threshold value	Flow birthrate factor, B_{list}	Flow space, S_{list}	Packet classification factor, C_{list}	Application factor[a], A_{list}
dec1				
2%	3,2%at 3122s	5,9%	31,7%	156
10%	9,2% at 3518s	14,9%	64,0%	783
dec2				
2%	2,4% at 3594s	6,7%	36,1%	150
10%	11,8% at 3561s	20,6%	70,8%	748
dec3				
2%	3,4% at 3517s	5,8%	40,5%	242
10%	10,7% at 3556s	13,4%	60,3%	1214
ebb900				
2%	28,4% at 3556s	38,5%	74,57%	184
10%	42,8% at 3591s	52,5%	82,8%	921
ebb115				
2%	18,7% at 3591s	16,65%	73,8%	172
10%	32,2% at 3657s	29,2%	85,9%	861
ebb130				
2%	14,0% at 3573s	13,9%	78,3%	153
10%	27,8% at 3600s	25,2%	86,5%	768

[a]Since the application factor is evident we present the number of applications detected

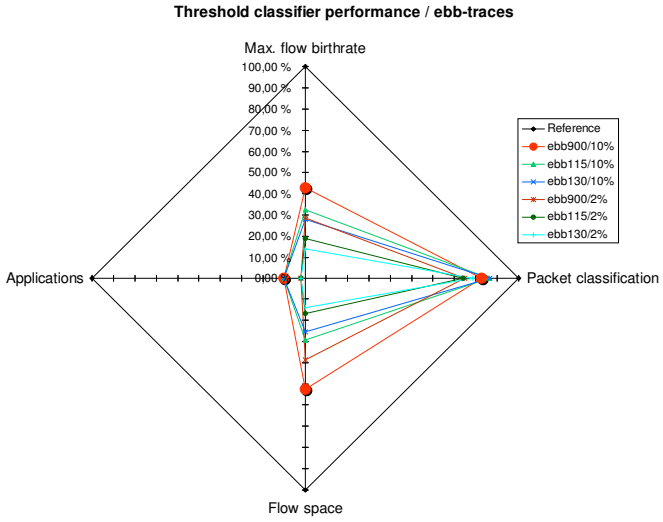
unrealistic aspect. First, the network application profile is determined after the flow analysis is performed. We use the measurement results of the past to determine a network application profile and then we use the obtained profile to the same trace. To compensate, the network application profiles from dec1 and ebb900 -traces are used as preset network application profiles for the rest of the dec- and for the rest of the ebb-traces respectively. On these occasions, we see an indication how the threshold classifier would perform in a more realistic fashion. The results for the threshold classifier with preset application lists are shown in Table 4.13 and Figure 4.13.

First, the application factor is somewhat lower in all of the traces due to the use of an application list from another network. We see a slight increase in the flow birthrate factor and use of flow space for the ebb-traces compared to Table 4.12, however, in dec-traces the flow birthrate factor and the use of the flow space is decreased compared to Table 4.12. Observing the packet classification factor in all of the traces, we see a decrease in the number of packets classified to priority. The decrease is significant in the dec-traces with the 2%-classifier showing a remarkably low values for the packet classification factor. The nature of the classifier behavior seems to be characteristic to a particular network environment and is therefore unpredictable over different networks.

The overall conclusion on the threshold classifier indicates that the performance is relatively stable and predictable in a given network environment. However, between the network environments there seems to be no clear dependency. Also the update of the



(a) dec-traces



(b) ebb-traces

Figure 4.12: Performance of the threshold classifier

priority application list should be frequent to ensure keeping up with the changes in the application profile.

4.4.6.2 Content performance analysis

Tables 4.14 and 4.15 present the content analysis figures. We can see that with the threshold classifier the coefficient of variation has not that much difference between classes as it does with the static application classifier. With the threshold classifier the difference between the coefficient of variation between classes 0 and 1 is slightly less than an order of magnitude whereas with the static application classifier the difference between classes was clearly in an order of magnitude. The difference in this regard is not big but noticeable. When comparing the priority class statistics with the threshold classifier and

Table 4.13: Performance results for the threshold classifier with preset list

Basic simulation results for the threshold classifier with a preset list				
Application threshold value	Flow birthrate factor, B_{list}	Flow space, S_{list}	Packet classification factor, C_{list}	Application factor, A_{list}
	B_{c-list}	S_{c-list}	C_{c-list}	A_{c-list}
dec1 application list				
dec2				
2%	2,1% at 3491s	5,5%	9,0%	1,12%/84
10%	11,7% at 3560s	18,3%	54,7%	9,7%/727
dec3				
2%	1,3% at 3511s	2,4%	3,1%	1,44%/175
10%	9,4% at 3546s	13,2%	42,0%	5,8%/715
ebb900 application list				
ebb115				
2%	27,4% at 3591s	62,9%	50,48%	1,3%/124
10%	42,2% at 3583s	76,0%	70,4%	6,6%/565
ebb130				
2%	10,0% at 3606s	14,1%	73,8%	1,04%/80
10%	40,5% at 3581s	33,7%	83,5%	9,1%/696

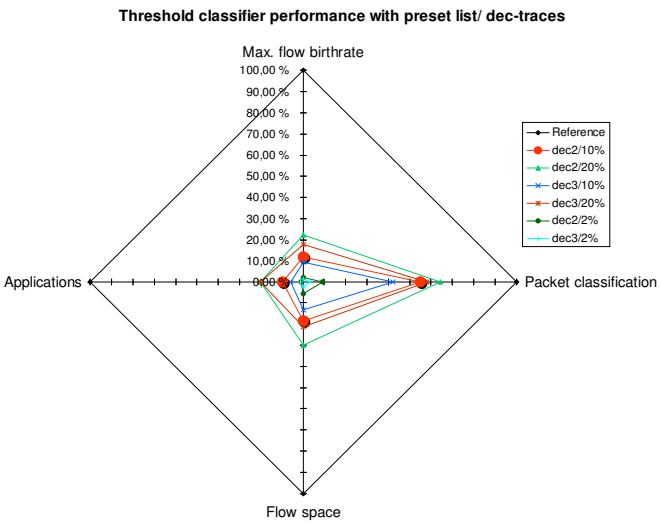
static classifier we can see that the coefficient of variation of the 2% threshold classifier for the classified packets and flows is consistently larger than that of the static classifier. With the 10% threshold classifier the difference is even more noticeable. Therefore, we claim that the threshold classifier has difficulties in detecting applications that behave differently. This statement holds regardless of the threshold setting or the network environment. Most probably the threshold classifier picks up "noise" in the form of applications that happen to lay around (and within) the threshold.

The default class statistics show roughly equal values of the coefficient of variation when compared to the static application classifier.

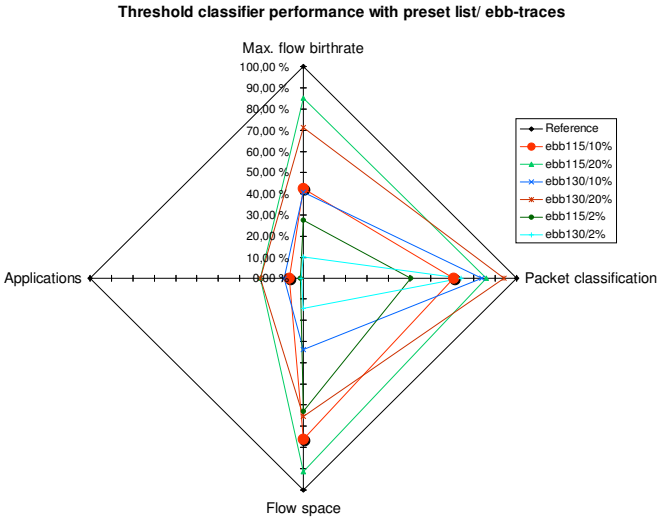
In Figures 4.14 to 4.19 the locations of the applications classified to higher priority are shown in the packet-flows –space. We can see that as we increase the threshold border from 2% to 10%, more applications in the lesser packets – more flows area are classified. This is in accordance with the defined selection criteria (high packet-flow –ratio) of the list classifier.

Looking at the mean of the classes and the suggested variation range of the classes we can see that in all network environments the classes are quite near to each other and the classes seem to behave in a similar manner.

In Tables 4.16 and 4.17 we see the applications classified to higher priority in the well-known port area (from 0 to 1023). The application lists contain a lot of the interactive ones such as news-service in 119/tcp, ssh and telnet in 22/tcp and 23/tcp and www-service in 80/tcp. However, the strictness of the border is easily observed in the dec-environment where in the dec1–trace we fail to detect any applications in the well-known ports. This phenomenon would be typical to a list classifier with a static threshold border.



(a) dec-traces



(b) ebb-traces

Figure 4.13: Performance of the threshold classifier with a preset application list

If the usage of a particular application diminishes, even temporarily and in relation to the measurement period, it is dropped out of the list.

4.5 Conclusions and the problem statement

We have discussed and briefly simulated some of the previously introduced traffic classification methods. Several suggestions have been made on different traffic classification schemes. These have been usually presented in the context of different flow classification methods. These studies mostly imply that traffic classification should direct the flow classification to optimize the performance of the IP router. No emphasis, however, is put to study the actual effect that these traffic classification methods have on the list of pri-

Table 4.14: Content statistics for dec-traces with threshold classification

Content statistics for dec-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
dec1/ 2%	Class 1 pkt	0,122%	0,03%	0,02%	1,49%	2
dec1/ 2%	Class 1 flw	0,002%	0,0003%	0,0006%	0,01%	1
dec1/ 2%	Class 0 pkt	0,01%	0,224%	0,00003%	18,63746%	27
dec1/ 2%	Class 0 flw	0,01%	0,526%	0,0006%	55,59726%	51
dec1/ 10%	Class 1 pkt	0,0736%	0,1395%	0,0029%	13,3956%	7
dec1/ 10%	Class 1 flw	0,0051%	0,01%	0,0006%	0,9936%	9
dec1/ 10%	Class 0 pkt	0,006%	0,177%	0,00003%	18,63746%	39
dec1/ 10%	Class 0 flw	0,0136%	0,526%	0,0006%	55,59726%	51
dec2/ 2%	Class 1 pkt	0,164%	0,058%	0,01%	5,55%	3
dec2/ 2%	Class 1 flw	0,003%	0,0006%	0,0006%	0,03%	3
dec2/ 2%	Class 0 pkt	0,01%	0,19%	0,00003%	12,2226%	23
dec2/ 2%	Class 0 flw	0,01%	0,46%	0,0006%	47,5911%	43
dec2/ 10%	Class 1 pkt	0,0853%	0,151%	0,0029%	11,11236%	7
dec2/ 10%	Class 1 flw	0,0074%	0,02%	0,0006%	2,26943%	12
dec2/ 10%	Class 0 pkt	0,0054%	0,122%	0,00003%	12,2226%	30
dec2/ 10%	Class 0 flw	0,0140%	0,458%	0,0006%	47,5911%	44
dec3/ 2%	Class 1 pkt	0,11%	0,07%	0,0004%	7,811%	4
dec3/ 2%	Class 1 flw	0,004%	0,19%	0,0003%	0,46%	6
dec3/ 2%	Class 0 pkt	0,006%	0,004%	0,00002%	16,01854%	31
dec3/ 2%	Class 0 flw	0,008%	0,40%	0,0003%	42,93669%	49
dec3/ 10%	Class 1 pkt	0,0424%	0,11%	0,0007%	9,17038%	8
dec3/ 10%	Class 1 flw	0,0042%	0,01%	0,0003%	0,95347%	7
dec3/ 10%	Class 0 pkt	0,0044%	0,168%	0,00002%	16,01854%	40
dec3/ 10%	Class 0 flw	0,0087%	0,40%	0,0003%	42,93669%	49

ority applications and, subsequently, to the user of the network. Our observations have been on router performance and the content and statistical properties of the classified applications. Following results have been obtained both in this and previously published work:

- All of the simulated classifiers reduce the use of router resources. In the connection oriented networks, this indicates that some traffic classification is useful if the connection setup load is to be kept within reasonable limits. The use of the flow space and the flow birthrate factor vary according to network environment.
- By using network based traffic classification with a static application set, use of the router resources is reduced compared to the reference classifier. In the static application classifier a static set of TCP/UDP–ports is selected to be classified to higher priority. This restricts, however, the application profile that evidently changes according to traffic fluctuations that occur in time and on network location. On the same note, by using a limited and static set of applications we lose the ability to quickly respond to the network application profile changes. Consequently, if the application set picked out is larger we see the problems shift into the problems

Table 4.15: Content statistics for ebb-traces with threshold classification

Content statistics for ebb-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
ebb900/ 2%	Class 1 pkt	0,399%	0,233%	0,001%	18,1552%	4
ebb900/ 2%	Class 1 flw	0,06%	0,042%	0,0014%	3,679%	5
ebb900/ 2%	Class 0 pkt	0,003%	0,08%	0,00009%	5,65%	28
ebb900/ 2%	Class 0 flw	0,01%	0,178%	0,0014%	11,2%	18
ebb900/ 10%	Class 1 pkt	0,087909%	0,235%	0,00073%	18,1552%	8
ebb900/ 10%	Class 1 flw	0,024076%	0,068%	0,0014%	4,8419%	9
ebb900/ 10%	Class 0 pkt	0,002286%	0,074%	0,0001%	5,6461%	34
ebb900/ 10%	Class 0 flw	0,009386%	0,17%	0,0014%	11,1828%	19
ebb115/ 2%	Class 1 pkt	0,41%	0,21%	0,0002%	13,5456%	4
ebb115/ 2%	Class 1 flw	0,04%	0,02%	0,0015%	0,81%	3
ebb115/ 2%	Class 0 pkt	0,003%	0,08%	0,0001%	6,09%	25
ebb115/ 2%	Class 0 flw	0,01%	0,18%	0,0015%	11,5094%	17
ebb115/ 10%	Class 1 pkt	0,093588%	0,215%	0,001%	13,5455%	7
ebb115/ 10%	Class 1 flw	0,021076%	0,057%	0,0015%	4,9963%	9
ebb115/ 10%	Class 0 pkt	0,002493%	0,0668%	0,0001%	6,0945%	29
ebb115/ 10%	Class 0 flw	0,010556%	0,174%	0,0015%	11,5094%	18
ebb130/ 2%	Class 1 pkt	0,4989%	0,247%	0,00027%	16,4349%	4
ebb130/ 2%	Class 1 flw	0,038%	0,016%	0,00186%	0,746%	3
ebb130/ 2%	Class 0 pkt	0,003%	0,073%	0,00008%	5,9138%	26
ebb130/ 2%	Class 0 flw	0,013%	0,202%	0,00186%	12,9275%	18
ebb130/ 10%	Class 1 pkt	0,108163%	0,25%	0,00163%	16,4349%	8
ebb130/ 10%	Class 1 flw	0,015355%	0,04%	0,00186%	3,5633%	9
ebb130/ 10%	Class 0 pkt	0,002433%	0,06%	0,00008%	5,9138%	30
ebb130/ 10%	Class 0 flw	0,012757%	0,20%	0,00186%	12,9275%	18

Table 4.16: A selection of priority applications picked out with the threshold classifier - dec-traces

Applications in different dec-networks with 2%-list threshold	
dec1	Total of 157 applications, none in well-known port numbers.
dec2	520/udp, 210/tcp, total of 150 applications.
dec3	520/udp, 28/udp, 1/udp, total of 242 applications.
Applications in different dec-networks with 10%-list threshold	
dec1	20/tcp, 119/tcp, 179/tcp, 514/udp, 520/udp (Total of 783 applications)
dec2	20/tcp, 119/tcp, 179/tcp, 210/tcp, 514/tcp, 520/udp (Total of 748 applications)
dec3	0/udp, 1/udp, 20/tcp, 28/udp, 69/tcp, 100/udp, 119/tcp, 179/tcp, 210/tcp, 514/udp, 520/udp, 801/udp, 899/tcp, 1000/udp, 1006/tcp (Total of 1214 applications)

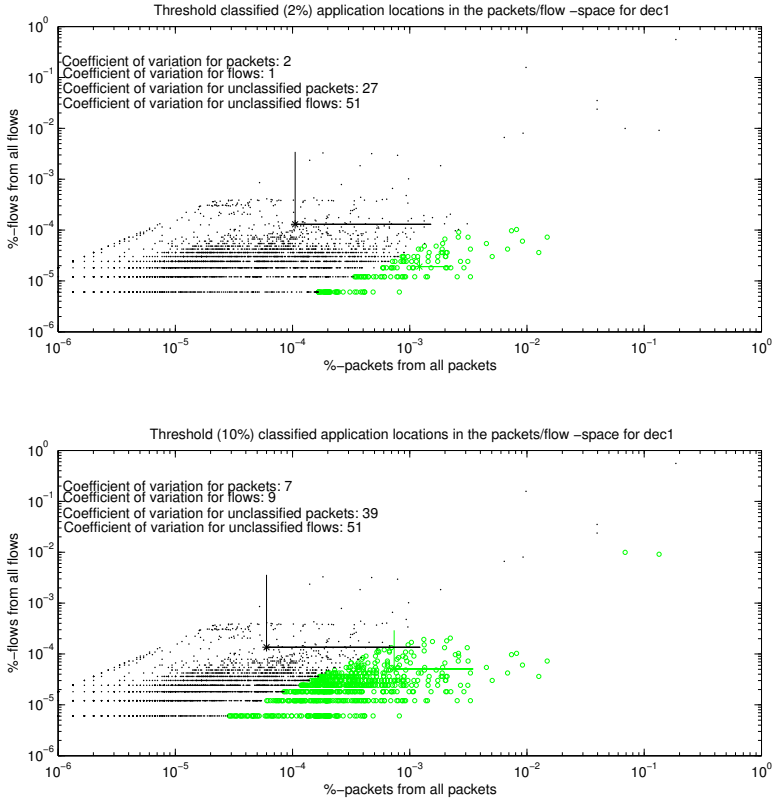


Figure 4.14: Locations of the priority applications using the threshold classifier / dec1

when all flows would be classified, i.e., the heavy use of resources and difficulty of realizing any meaningful quality differentiation. Static application classification can be set to pinpoint the familiar and well-known applications, but requires careful supervision if it is to react to changes in the network application profile.

- The threshold classifier can be adjusted to detect a certain amount of applications that produce the majority of packets in the network. This classification method seems to use the router resources in an efficient manner in a connection oriented environment. Due to the static border it is, however, sensitive to the smallest of changes in the use of applications. If optimizing network performance, the value for list threshold could be fed back with the information on the use of resources. The application profile of the threshold classifier seems to contain a wide scope of different types of applications and the threshold classifier does not seem to be picking up a particularly consistent behavior in the application set.

- The threshold classifier would seem to suit networks where the policy would be towards optimizing the packet forwarding and flow birthrate -factors. With reasonable flow birthrate one would be able to forward a large amount of packets. However, since the prioritized applications include applications of varying behavior and nature, the threshold classifier would not be the optimal choice if user satisfaction would be the goal of the network policy.

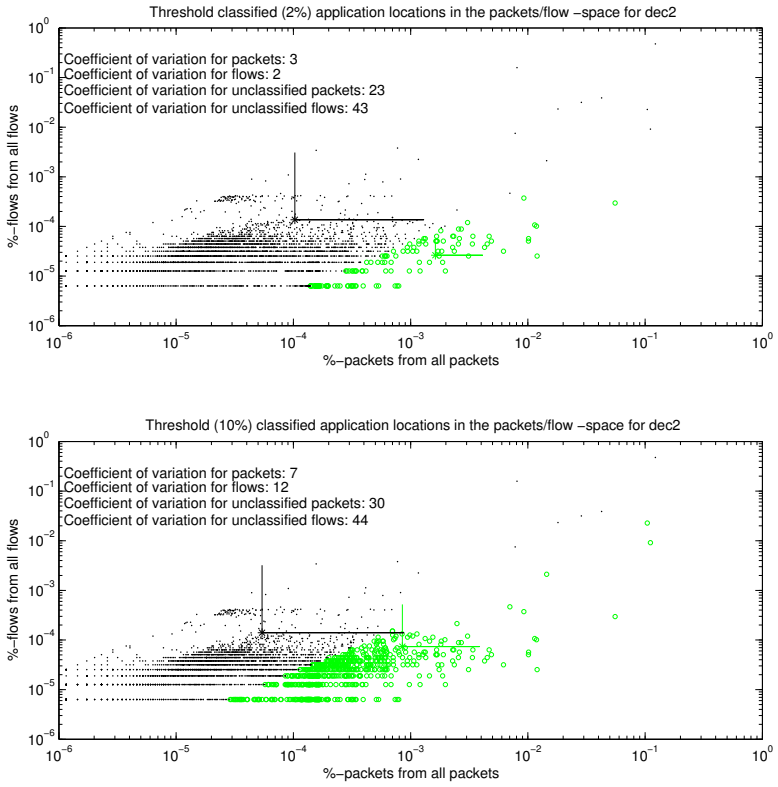


Figure 4.15: Locations of the priority applications using the threshold classifier / dec2

- The threshold classifier supports only one rigid policy. This is true even if we make the threshold limit dynamic and try to optimize router resources like in the dynamic packet count classifier. The network manager may want to treat traffic differently for applications that produce especially high amount of traffic or particularly low amount of traffic or especially high or low amount of flows or a small number of applications that are on the boundary. Therefore a "non-linear" solution would be more flexible and thus preferable. The threshold classifier is linear in the sense that in the packet-flow –space the boundary in the threshold classifier is always a straight line (cf. Figs. 4.14 and 4.19). A non-linear solution would allow the boundary to be convex or concave, squiggly or even separate areas from the packet-flow –space.

When observing the locations of the chosen applications in the packet-flow –space, we notice that some of the more interesting and interactive applications tend to lie in the high packet count - low flow count area. This raises a question on if it would be possible somehow to adapt to traffic profile changes by detecting the areas where different application types lie. It would be particularly interesting to use samples of measurement results to indicate areas in the packet-flow –space for different types of applications.

Although the classification methods presented in this chapter can optimize the network element (router) performance in an efficient manner, it seems that the current implemen-

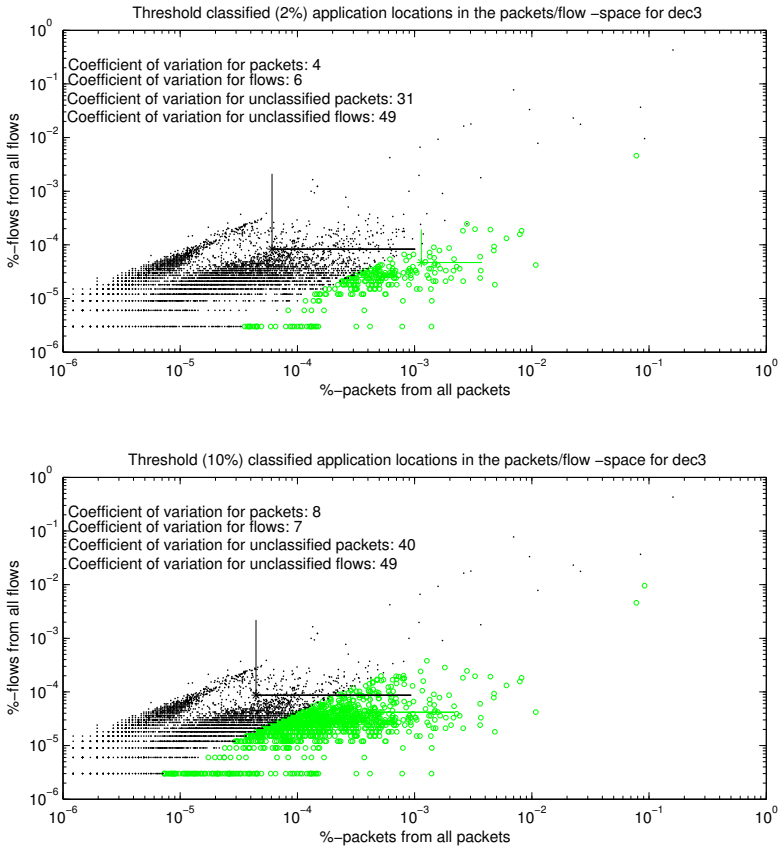


Figure 4.16: Locations of the priority applications using the threshold classifier / dec3

tations of traffic classification schemes do not offer means to provide adequate content performance. This is why we will seek to introduce a classification scheme that, in addition to adequate context performance, would also be able to offer a possibility to detect applications and create policies that also offer the user improved content performance and that this improved performance is directed to those applications that naturally are expected to perform well by the user (interactive and popular applications such as WWW, Telnet, X-win, ftp, etc.). The application lists that are chosen based on the measurements should reflect the changes in the traffic profile and in the behavior of individual applications. The primary interest should be on the quality of the application profile, although we should control and measure the use of resources also. Taking into account the scalability issues, the classification scheme should also be of use in an aggregated traffic environment where individual traffic flows might be grouped together. All in all, the policy used in the network should always be a combination of a dynamic and a static component. The latter would contain network control traffic (name services etc.) and other applications hand picked by the network manager to a certain class, whereas the former would be formed with the aid of classification methods that are able to detect different types of applications from the network traffic mix.

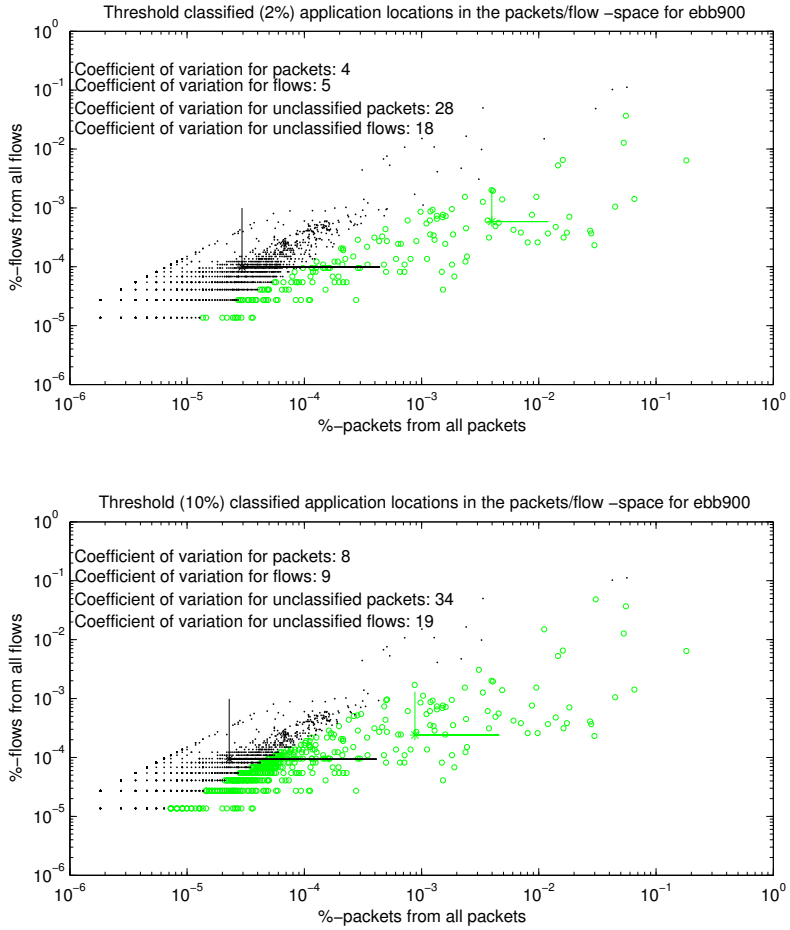


Figure 4.17: Locations of the priority applications using the threshold classifier / ebb900

4.6 Summary

In this chapter we gave a general overview of different, previously introduced, methods of traffic classification. The performance analysis of a traffic classification scheme is done by observing the effect that this scheme would have on the forwarding components of the network and the effect on the user. Starting from the next chapter this work will concentrate on developing a new classification method based on analysis of traffic measurements.

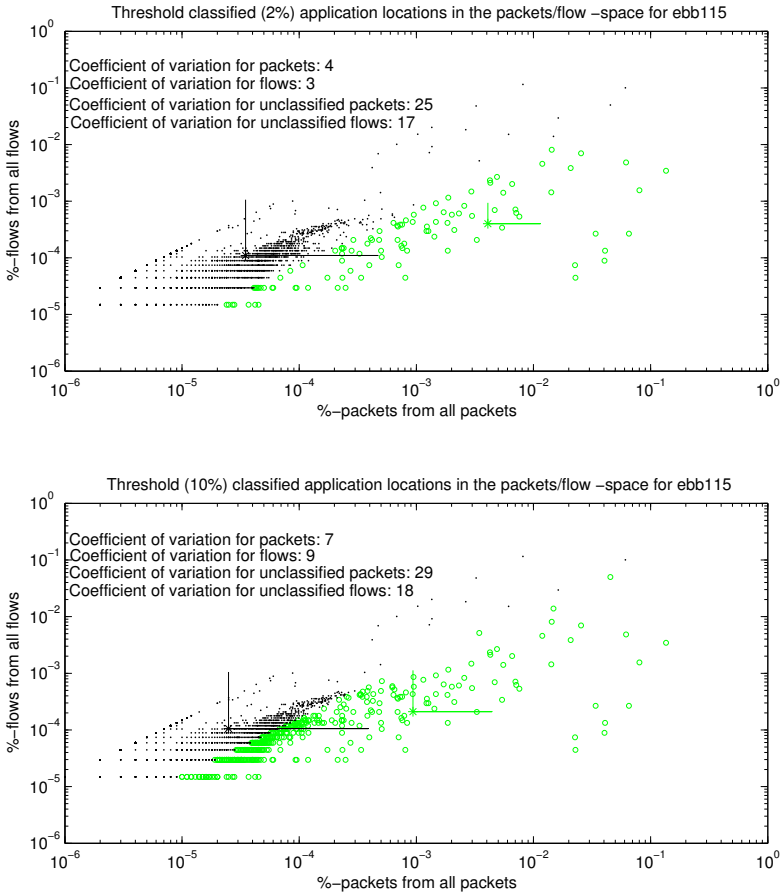


Figure 4.18: Locations of the priority applications using the threshold classifier / ebb115

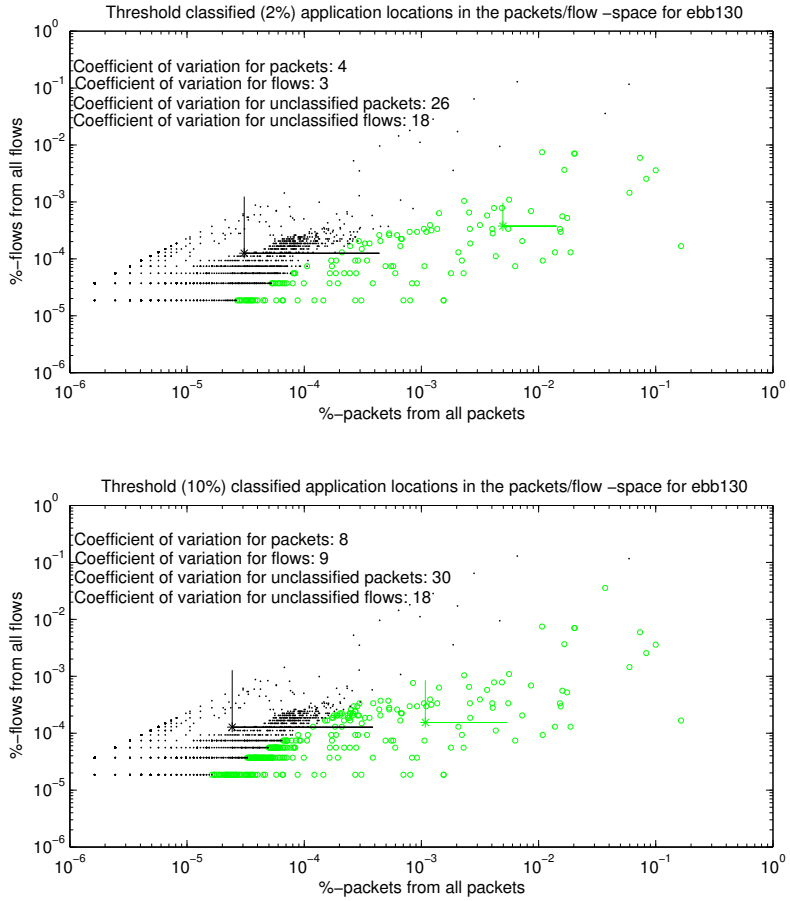


Figure 4.19: Locations of the priority applications using the threshold classifier / ebb130

Table 4.17: A selection of priority applications picked out with the threshold classifier - ebb-traces

Applications in different networks with 2% list threshold	
ebb900	0/udp, 1/udp, 119/tcp, 82/udp, 20/tcp, 22/tcp, 23/tcp, 513/tcp, 600/tcp, 601/tcp, 602/tcp, 801/udp, 1019/tcp, total of 184 applications
ebb115	0/udp, 20/tcp, 68/udp, 119/tcp, 25/tcp, 515/tcp, 52/udp, 33/udp, 22/tcp, 23/tcp, total of 172 applications.
ebb130	20/tcp, 23/tcp, 33/udp, 68/udp, 79/tcp, 22/tcp, 119/tcp, 139/tcp, , total of 154 applications.
Applications in different networks with 10% list threshold	
ebb900	0/udp, 1/udp, 2/udp, 20/tcp, 22/tcp, 23/tcp, 52/udp, 53/tcp, 68/udp, 73/udp, 79/tcp, 82/udp, 119/tcp, 130/udp, 139/tcp, 213/udp, 231/udp, 250/udp 513/tcp, 600/tcp, 601/tcp, 602/tcp, 801/udp, 897/tcp, 995/tcp, 998/tcp, 1008/tcp, 1015/tcp, 1019/tcp, 1020/tcp, 1021/tcp, 1022/tcp, 1023/tcp (Total of 921 applications)
ebb115	0/udp, 20/tcp, 21/tcp, 22/tcp, 23/tcp, 25/tcp, 33/udp, 52/udp, 53/tcp, 68/udp, 79/tcp, 119/tcp, 139/tcp, 213/udp, 389/tcp, 513/tcp, 515/tcp, 600/tcp 719/udp, 794/tcp, 796/tcp, 973/udp, 988/tcp, 1019/tcp, 1021/tcp, 1022/tcp, 1023/tcp (Total of 861 applications)
ebb130	0/udp, 20/tcp, 22/tcp, 23/tcp, 33/udp, 52/udp, 53/tcp, 57/udp, 68/udp, 79/tcp, 119/tcp, 139/tcp, 213/udp, 513/tcp, 520/udp, 618/udp, 708/udp, 801/udp 901/udp, 1004/tcp, 1007/tcp, 1010/tcp, 1015/tcp, 1017/tcp, 1019/tcp, 1020/tcp, 1021/tcp, 1022/tcp, 1023/tcp (Total of 768 applications)

Chapter 5

The 2-class classification based on flow analysis and the LVQ algorithm

An extension to the static application list based classification scheme would be a solution where the list of prioritized applications is updated regularly [9]. These updates are based on the traffic characteristics measured from the network traffic. The basic idea in a traffic measurement based classifier is to measure some property or properties of the network traffic as a background, non-real-time process, and then set the criteria for packet classification, classification rules, according to the traffic statistics. For instance, we could measure an application that sends a large number of packets with short and regular spacing between packets, and classify such an application to be of a streaming kind.

In Chapter 3 we discussed how the flow count gives an indication how bursty and fragmented the sending of the packets by the application is. In this chapter we will first validate our discussion in Chapter 3 by observing the packet and flow counts of different applications. We will advance to examine how to apply the LVQ algorithm to the results of flow analysis in a traffic classifier that divides traffic to two classes. We then observe the content and context performance of such a classifier implemented in the simulation environment.

5.1 Flow and packet counts

At this point we introduce a new trace where the application types are more varied. The trace properties are shown in Table 5.1.

We start by performing flow analysis (cf. to Table 3.2) observing the amount of flows and packets by a selected set of applications compared to the total number of flows and packets. The applications are chosen so that they include some of the familiar applications like the usenet news (nntp, port 119), telnet (port 23), ssh (port 22) and www (http, port 80). In addition we also look at the protocols used for mail delivery (smtp at port 25 and imap at port 143). We have also included some network control

Table 5.1: Properties of the tct-trace

Trace information			
Name	*Location*	*Length*	*Nr of packets*
tct-heina-1999 (tct)	Access point of the Laboratory of Telecommunications Technology in Helsinki University of Technology	2998656 s (approximately 34 days and 17 hrs)	14853713

traffic like the domain name service (dns, port 53) and NETBIOS name service (port 137). Moreover, in the tct-trace we are observing the behavior of the flow count in an IP phone application (Vocaltech, port 22555) and a real-audio stream (port 5900). The complete list of selected applications is shown in Table 5.2.

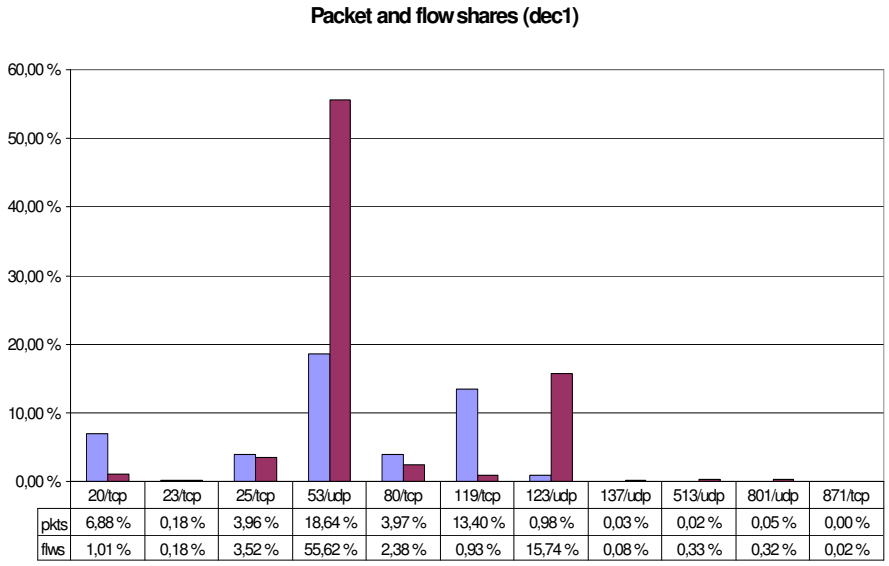
Table 5.2: The selected application set for the application characterization

Application set $\mathbb{S}_{SAC}$	
Port number and protocol($Sport$)/($Proto$)	Application description[a]
20/tcp	ftp default data
22/tcp	SSH remote login
23/tcp	Telnet
25/tcp	Simple Mail Transfer
53/udp	DNS, name service
79/udp	Finger
80/tcp	WWW-protocol, HTTP
119/tcp	Network News protocol, NNTP
123/tcp	Network time protocol
137/udp	NETBIOS name service
143/udp	Internet Message Access Protocol
513/udp	who, maintains databases showing who's logged in to machines on a local net and the load average of the machine
801/udp	device
871/tcp	unassigned
5900/tcp	unassigned, used for RealAudio
22555/udp	Vocaltec Internet Phone

[a] As in http://www.iana.org/assignments/port-numbers

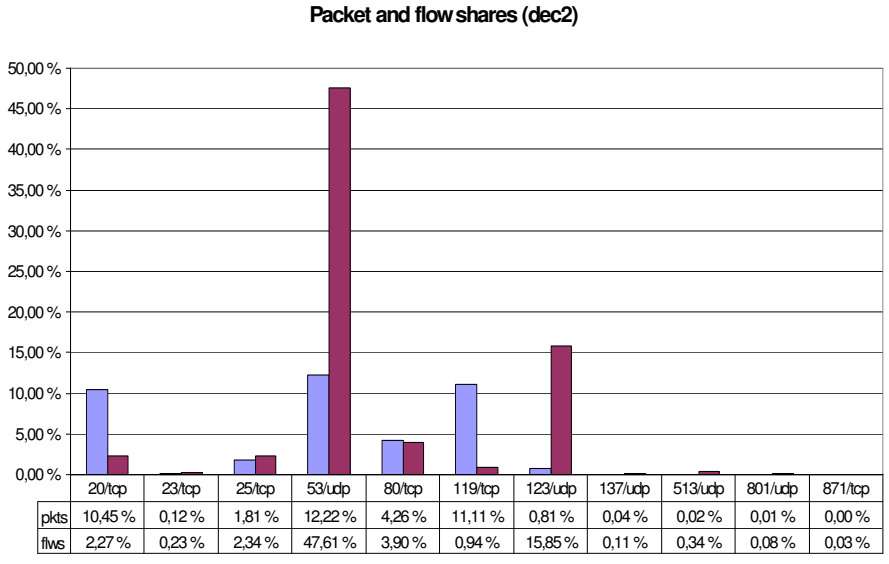
Packet count values in Figures 5.1 to 5.5 show the packet count of the packets belonging to the selected applications as compared to all the packets in the trace. Flow count values in Figures 5.1 to 5.5 show the flow count of the applications compared to all the flows in the trace.

We can see, looking at the packet and flow shares, that individual applications have distinct features. For instance, the dns-service seems to always create significantly more



	20/tcp	23/tcp	25/tcp	53/udp	80/tcp	119/tcp	123/udp	137/udp	513/udp	801/udp	871/tcp
pkts	6,88 %	0,18 %	3,96 %	18,64 %	3,97 %	13,40 %	0,98 %	0,03 %	0,02 %	0,05 %	0,00 %
flws	1,01 %	0,18 %	3,52 %	55,62 %	2,38 %	0,93 %	15,74 %	0,08 %	0,33 %	0,32 %	0,02 %

(a) dec1

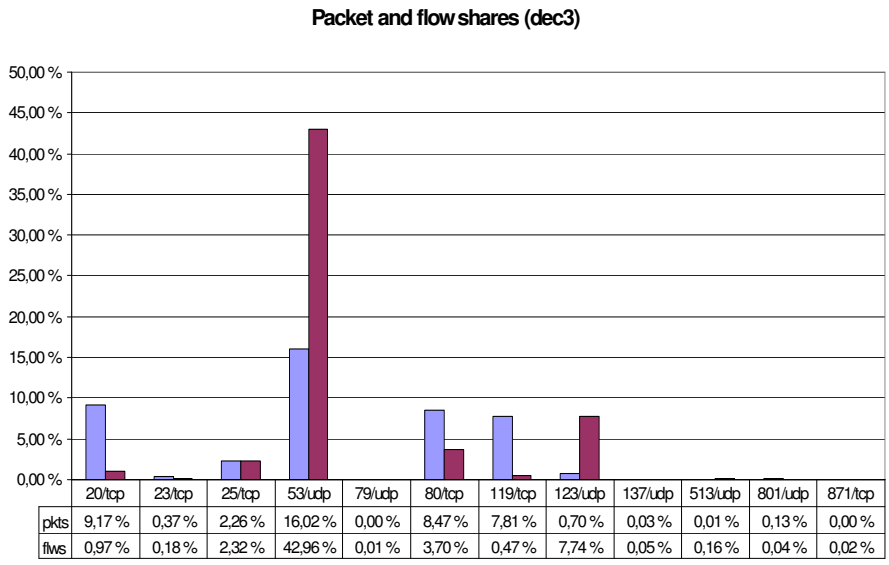


	20/tcp	23/tcp	25/tcp	53/udp	80/tcp	119/tcp	123/udp	137/udp	513/udp	801/udp	871/tcp
pkts	10,45 %	0,12 %	1,81 %	12,22 %	4,26 %	11,11 %	0,81 %	0,04 %	0,02 %	0,01 %	0,00 %
flws	2,27 %	0,23 %	2,34 %	47,61 %	3,90 %	0,94 %	15,85 %	0,11 %	0,34 %	0,08 %	0,03 %

(b) dec2

Figure 5.1: Packet and flow shares in dec1 and dec2 (darker/red for the flows, grey/blue for the packets)

flows per packet than any other application. Dns also seems to be the dominant application, except in the tct-trace, but this is expected since many applications use the dns-service. The applications with direct user interaction, such as telnet, ssh and usenet news (ports 23, 22, and 119) create very few flows per number of packets, but, if observed together as a group, create most of the packets. The www-service behaves ambiguously. In dec-traces the www-service has not yet made its breakthrough and its share of packets and flows is on the minority. In ebb-traces and in the tct-trace the www-service creates the majority of packets, being second only to the news-service in ebb115. As the share of packets has increased so has the flow count. In ebb-traces and in the tct-trace the



	20/tcp	23/tcp	25/tcp	53/udp	79/udp	80/tcp	119/tcp	123/udp	137/udp	513/udp	801/udp	871/tcp
pkts	9,17 %	0,37 %	2,26 %	16,02 %	0,00 %	8,47 %	7,81 %	0,70 %	0,03 %	0,01 %	0,13 %	0,00 %
flws	0,97 %	0,18 %	2,32 %	42,96 %	0,01 %	3,70 %	0,47 %	7,74 %	0,05 %	0,16 %	0,04 %	0,02 %

Figure 5.2: Packet and flow shares in de3 (darker/red for the flows, grey/blue for the packets)

www-service is responsible for creating a significant portion of the flow count; a phenomenon reported also in [19, 72]. This is somewhat expected based on the application behavior: Relatively short documents or parts thereof, possibly also from various www-servers, are sent with the TCP-protocol leading to a situation where the data transfers are happening in the 'slow start' -phase of the TCP connection. The use of later versions of HTTP-protocol and its *pipelining* and *persistent connections* features may help to use the TCP–protocol more optimally and reduce the flow count [73].

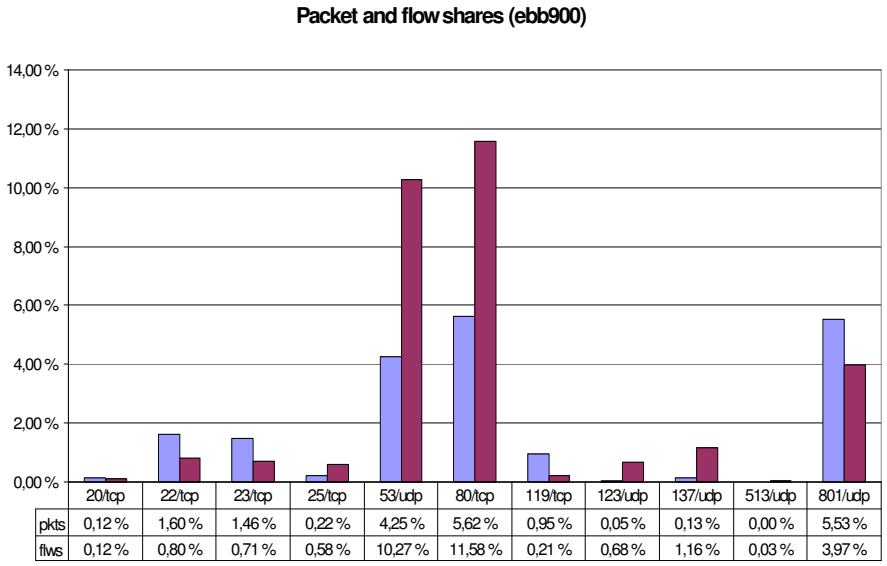
5.1.1 Conclusions for traffic classification

Distinguishing applications based solely on the flow or packet count alone is not straightforward and is subject to misclassifications. The packet streams of large quantity are often created by short requests from the user. The high count of packets indicates the amount of data that a user has to send. The low flow count, on the other hand, suggests an application that sends its packets on a relatively high rate (lots of packets spaced within the flow timeout value) for a long time. Using the flow count to observe the burstiness and packet count to observe for application presence in the network, traffic could be separated into four classes with relative ease.

For the number of traffic categories it is stated in [74] that the Internet today carries three basic categories of traffic, and any QoS environment must recognize and adjust itself to these three basic categories. The categories are:

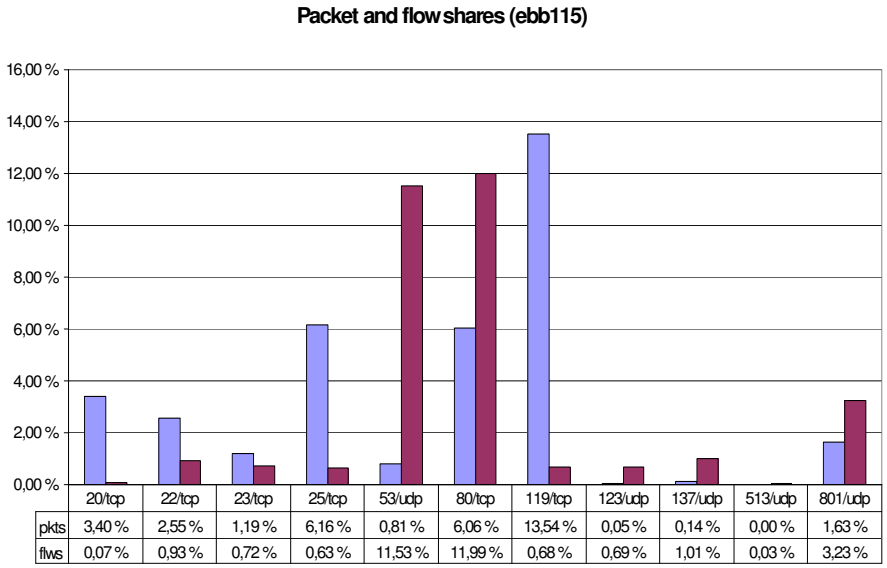
1. The long held adaptive reliable traffic flows, typically including the long held TCP traffic flows.

2. The short duration reliable transactions, where the lifetime of the traffic flow is that short that the flow sits completely within the startup phase of the TCP adaptive



	20/tcp	22/tcp	23/tcp	25/tcp	53/udp	80/tcp	119/tcp	123/udp	137/udp	513/udp	801/udp
pkts	0,12 %	1,60 %	1,46 %	0,22 %	4,25 %	5,62 %	0,95 %	0,05 %	0,13 %	0,00 %	5,53 %
flws	0,12 %	0,80 %	0,71 %	0,58 %	10,27 %	11,58 %	0,21 %	0,68 %	1,16 %	0,03 %	3,97 %

(a) ebb900



	20/tcp	22/tcp	23/tcp	25/tcp	53/udp	80/tcp	119/tcp	123/udp	137/udp	513/udp	801/udp
pkts	3,40 %	2,55 %	1,19 %	6,16 %	0,81 %	6,06 %	13,54 %	0,05 %	0,14 %	0,00 %	1,63 %
flws	0,07 %	0,93 %	0,72 %	0,63 %	11,53 %	11,99 %	0,68 %	0,69 %	1,01 %	0,03 %	3,23 %

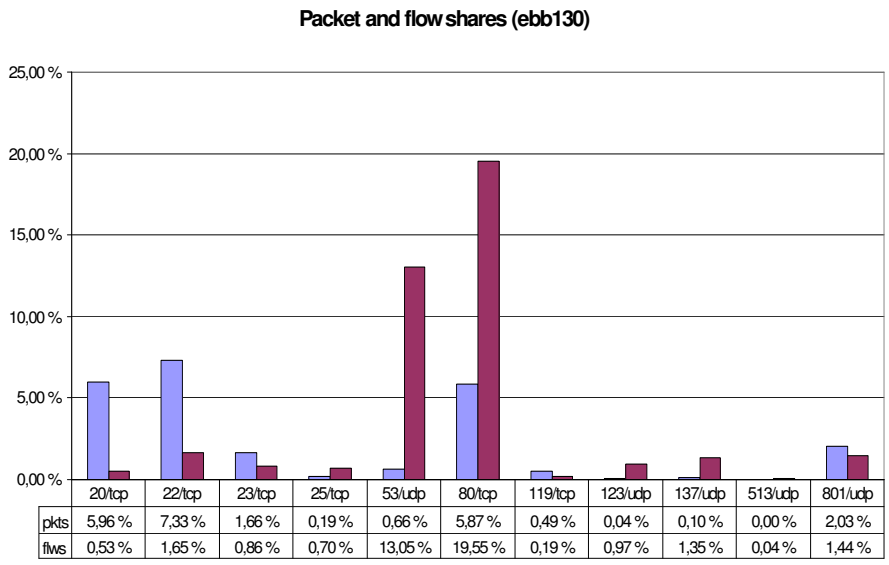
(b) ebb115

Figure 5.3: Packet and flow shares in ebb900 and ebb115 (darker/red for the flows, grey/blue for the packets)

flow control protocol.

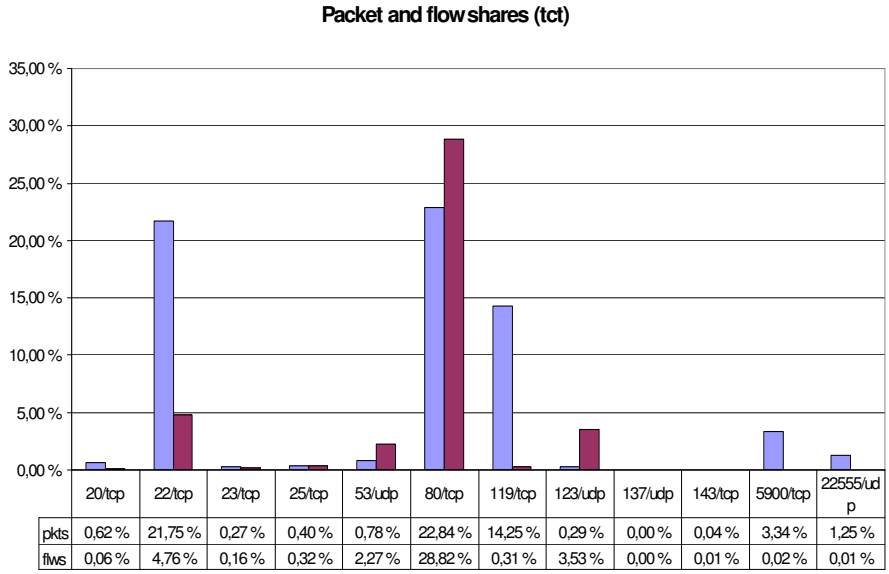
3. The externally controlled unidirectional traffic flows, which are typically a result of compression of a real time audio or video data.

Furthermore in [75], it is stated that the traffic should be divided into connection-oriented and connectionless applications and it is stated that these two types of applications is a necessity for future networks.



	20/tcp	22/tcp	23/tcp	25/tcp	53/udp	80/tcp	119/tcp	123/udp	137/udp	513/udp	801/udp
pkts	5,96 %	7,33 %	1,66 %	0,19 %	0,66 %	5,87 %	0,49 %	0,04 %	0,10 %	0,00 %	2,03 %
flws	0,53 %	1,65 %	0,86 %	0,70 %	13,05 %	19,55 %	0,19 %	0,97 %	1,35 %	0,04 %	1,44 %

Figure 5.4: Packet and flow shares in ebb130 (darker/red for the flows, grey/blue for the packets)



	20/tcp	22/tcp	23/tcp	25/tcp	53/udp	80/tcp	119/tcp	123/udp	137/udp	143/tcp	5900/tcp	22555/udp
pkts	0,62 %	21,75 %	0,27 %	0,40 %	0,78 %	22,84 %	14,25 %	0,29 %	0,00 %	0,04 %	3,34 %	1,25 %
flws	0,06 %	4,76 %	0,16 %	0,32 %	2,27 %	28,82 %	0,31 %	3,53 %	0,00 %	0,01 %	0,02 %	0,01 %

Figure 5.5: Packet and flow shares in tct-trace (darker/red for the flows, grey/blue for the packets)

In this work, for classifying traffic into two classes, we use the following class properties as detailed in Table 5.3 and aim to realize policy rules where applications with high presence and long bursts are given priority over other traffic types. This means that we aim to combine the long held traffic flows and the externally controlled traffic flows into one class and the short duration transaction traffic to another.

For every application we will measure the relative flow and packet counts and place the

Table 5.3: Traffic classes for 2-class classification

Service class properties for the 2-class LVQ classifier	
Class nr	Intended Class properties
Class 0	Low priority traffic, the rest of the applications
Class 1	Priority traffic, applications that have high and smooth presence.

results in the packet-flow –space as shown in Figure 3.2. The problem after this lies in determining the areas and area boundaries of different traffic types in the packet-flow –space.

5.1.2 Overview of the proposed solution

Our aim is to develop a classifier that works off-line and that is given a few classified examples on how to classify application flows based on packet and flow information. Since the amount of applications in the network, according to the IP application definition earlier, might be quite high, we look for a method to reduce the workload of classifying a large number of data samples in the packet-flow –space. Therefore, we need a method that is able to classify individual data vectors to a proper class with the aid of a relatively small set of pre-classified data vectors. In the following, we will design an off-line type of traffic classification method, where the packets and flows per application -relations are observed, and based on pre-classified examples the classifier detects applications of similar packet/flow -characteristics. The classifier will be built based on the Learning Vector Quantization (LVQ) algorithm [5].

Figure 5.6 shows how the application of LVQ is done. We start by collecting a packet trace and subjecting it to flow analysis. Then the packet and flow count (as defined in Chapter 3) results are input to the LVQ algorithm together with the teaching samples to produce the classification results. The classified application list is then input to flow classification (resembling an IP router performing packet classification) and the context performance and the content evaluation (as defined in Chapter 3) are then performed and the results are analyzed.

5.2 Learning Vector Quantization

So far, in Chapter 4, we have used the traffic measurements to determine the packet and flow counts per application. This has left us with, depending on the network characteristics, approximately 5000 to over 10000 data vectors containing the application identifier a, normalized packet count for the application according to (3.9) and normalized flow count for the application according to (3.10). To determine the appropriate classification decision for a new measurement result would require to classify all existing data vectors and then classify the unknown vector with k–nearest–neighbor (kNN) classification.

The kNN methods for classification have provided good performance on a variety of real-life data sets and often perform better than more complicated approaches. Two possible reasons are stated in [76]:

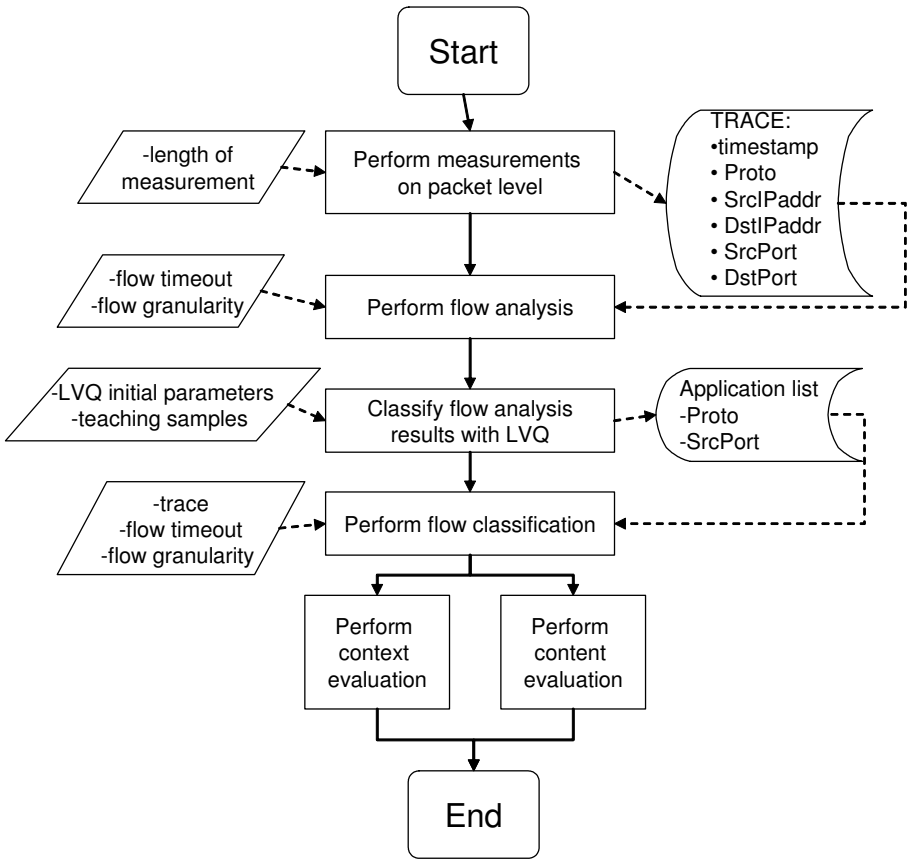


Figure 5.6: The process chart for the application and analysis in traffic classification with LVQ algorithm

1. Practical problems often have a low intrinsic dimensionality even though they may have many input variables. If some of the input variables are interdependent, the data lie on lower-dimensional manifold within the input space and this effectively reduces the dimensionality of the problem.

2. Accurate estimates of conditional probabilities are unnecessary for accurate classification. In some sense, the problem of classification is easier than that of regression, so the result of dimensionality is less severe.

For problems with many data samples, classifying a particular input vector using (kNN) methods poses, however, a large computational burden, since it requires storing and comparing all the samples. One way to reduce this burden is to represent the large dataset by a smaller number of prototype vectors. This approach requires a procedure for choosing these prototype vectors so that they provide high classification accuracy.

The solution provided by Kohonen in [5] is to use learning vector quantization (LVQ) methods to determine initial locations of prototype vectors, then assign class labels to these prototypes, and then adjust the locations using a heuristic strategy that tends to reduce the empirical misclassification risk [76].

The purpose of the LVQ algorithm is to define class regions in the input data space and it is therefore meant to be a method for statistical classification or recognition. To this end,

a subset of similarly labelled codebook vectors are placed into each class region. If the class distributions of the input samples would overlap at the class borders the codebook vectors of each class can be placed in and shown to stay within each class region for all times.

In the LVQ algorithms, vector quantization is not used to approximate the density functions of the class samples, but to directly define the class borders according to the nearest–neighbor rule. The accuracy of the classification and the time needed for learning using LVQ algorithms depends on

- an optimal (or near–optimal) number of codebook vectors assigned to each class and their initial values and
- proper learning rate and proper criterion for the stopping of learning.

Because of unknown forms of the class distributions, the final placement of the codebook vectors is unknown until at the end of the learning, their optimal numbers cannot be determined before that. Therefore, the assignment of codebook vectors to classes must be done gradually, or iteratively [5].

The LVQ algorithm belongs to the class of algorithms that use supervised learning. This means that the LVQ algorithm, by definition, needs pre-classified samples to function. One immediately asks whether there would be any unsupervised methods to aid in the classification process. Using one of the unsupervised methods, for example Self-Organizing Maps[5], we would be able to extract from the input samples groups or clusters of spatial neighbors. Clustering algorithms strongly rely on the natural grouping of the data and the proper initialization of the algorithm. Therefore, using unsupervised algorithms to support classification might easily lead to a situation where the number of classes would be either too high or too low, or the class boundaries would cross over different types of application classes. Consequently, although admittedly giving in our requirement for as automatic detection of traffic classes as possible, we have chosen to use an algorithm that needs supervision and pre-classified samples. This has given us the freedom to choose and limit the number of traffic classes appropriately. However, it should be noted that the unsupervised clustering algorithms might be used to find proper teaching samples for the supervised classification method. This is, however, left for future research.

5.2.1 Constructing the teaching set

The construction of the teaching set is the most important step to be taken when applying the LVQ algorithm to traffic classification. The teaching set largely determines the type of applications that are picked up to each class and indicates the network policy. Also, according to [25, 77], it is advantageous to divide the traffic according to its behavior and, therefore, we should pick the teaching samples so that they represent the different application behavior types found in the network. Therefore, the examples picked should represent the desired class behavior in the best possible way and our aim is to pick a selection of applications that have typical characteristics for certain application types as suggested in Table 5.3. Consequently, picking the teaching vectors is dictated by the policy that is carried out in the network and, therefore, teaching vectors should be selected carefully.

The teaching sample, x, has five parameters,

$$x =< Proto, Sport, C_{norm}(a), F_{norm}(a), c > \quad (5.1)$$

of which $C_{norm}(a)$ and $F_{norm}(a)$ are used in the LVQ algorithm and the application identifier and traffic class identifier are used for labelling in the classification process. The individual teaching vectors may be combined to a set of teaching vectors $\mathbb{X}$. We note, that the individual teaching samples indicate a neighbourhood where all other applications will be classified according to the sample. This makes it possible to choose various areas from the packet-flow –space for classification and thus better take into consideration the application characteristics as they present themselves in the packet-flow –space. Thus the division of applications into different classes is not linear as is the case with the list classifier presented in Chapter 4.

The application identifiers for teaching vectors for prioritized and default traffic are shown in Table 5.4.

Table 5.4: Application sets used to teach the LVQ classifiers

Pre-classified teaching samples, $\mathbb{X}$		
Network	Classification	Application list, *Sport*
dec1	Prioritized, $\mathbb{X}_{c=1}$	20/tcp, 23/tcp, 80/tcp, 119/tcp
	Default, $\mathbb{X}_{c=0}$	53/udp, 123/udp, 513/udp, 801/udp, 871/tcp
dec2	Prioritized, $\mathbb{X}_{c=1}$	20/tcp, 23/tcp, 80/tcp 119/tcp
	Default, $\mathbb{X}_{c=0}$	53/udp, 123/udp, 513/udp, 801/udp, 871/tcp
dec3	Prioritized, $\mathbb{X}_{c=1}$	20/tcp, 23/tcp, 80/tcp, 119/tcp, 513/tcp
	Default, $\mathbb{X}_{c=0}$	53/udp, 79/udp, 513/udp, 540/udp
ebb900	Prioritized, $\mathbb{X}_{c=1}$	22/tcp, 23/tcp, 80/tcp, 119/tcp
	Default, $\mathbb{X}_{c=0}$	53/udp, 123/udp, 137/udp, 513/udp
ebb115	Prioritized, $\mathbb{X}_{c=1}$	22/tcp, 23/tcp, 80/tcp, 119/tcp, 515/tcp
	Default, $\mathbb{X}_{c=0}$	37/udp 53/udp, 123/udp, 513/udp
ebb130	Prioritized, $\mathbb{X}_{c=1}$	20/tcp, 22/tcp, 23/tcp, 80/tcp, 119/tcp
	Default, $\mathbb{X}_{c=0}$	53/udp, 123/udp, 134/udp, 137/udp, 513/udp

We limit the selection of the teaching set to applications using port–numbers in the well-known port number area or other port numbers for well-established applications. For the classification process, we need to select applications for both priority and default treatment and we aim to pick out as even number as possible of prioritized and default application samples. The aim is to create policy rules that place a lot of packets on few flows. The idea is to have a relatively few flows to carry the majority of all packets. Therefore, the teaching vectors should be chosen from the high and smooth presence applications. It should be noted that this choice on policy rule type is arbitrary, presents just one option and may be modified according to the needs of the network operator.

The applications used for teaching the classifier are the best examples that we were able to determine without surrendering to too much guessing. For instance, the lack of any hard-interactive traffic in all of the traffic traces (e.g., 2-way Voice over IP) prevents us

from using these applications as teaching samples. Also, in the priority set, there are many applications that behave differently when compared with each other. We see in Table 5.4 that the applications that are used as teaching samples for the priority traffic class can be argued to be relatively interactive applications (telnet, www). On the other hand, applications used as examples for default traffic are more network control oriented. Consequently, if this kind of classification is applied in real networks, the decision to prioritize should accompany a suitable allocation of resources or an indication of proper service class. Furthermore, although the teaching vectors indicate that network control traffic should be treated as default traffic, it should be obvious that the classification procedure should not slow or stop the network from functioning because network control traffic is not prioritized.

Placing of the teaching samples in the packet-flow –space with $x(C_{norm}(a), F_{norm}(a))$ on the logarithmic scale is shown in Figures 5.7 to 5.10.

We can see from Figures 5.7 to 5.10 that the applications classified to higher priority seem to reside more in the high packet - low flow part of the packet-flow –space. As the number of flows of an application increases, the more likely the application is going to receive default classification and treatment. Figures 5.7 to 5.10 show that the policy rules we are creating aim to put applications with high and smooth presence to higher priority than other applications. We also note at this point, that in Figure 5.8 the dec3-trace seems to have one outlier in the prioritized teaching vectors. This vector is placed relatively low in the packet-dimension and has also low flow count.

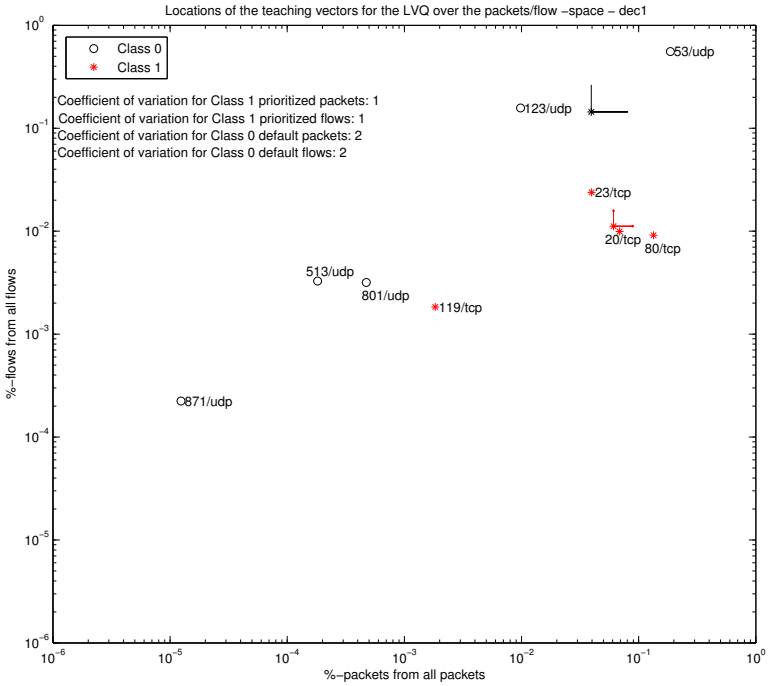
5.2.2 Codebook size

A codebook vector is a representative for several data vectors. As the codebook vector is taught it is given the class assignment according to the classifier construction process. Subsequently all of the data vectors are classified against the codebook vectors, not the pre-classified teaching vectors or other classified data vectors. The number of the codebook vectors projecting the pre-classified data needs to be big enough so that an adequate amount of vectors are placed on each clustering of applications in the packet-flow –space (refer to Figure 3.2). The amount of codebook vectors tells us to how many points in the decision space the teaching vectors are mapped. In addition, the number of codebook vectors should be large enough to present all the local packets/flow -clusters [5, 76]. In this work, the number of codebook vectors is set to 400.

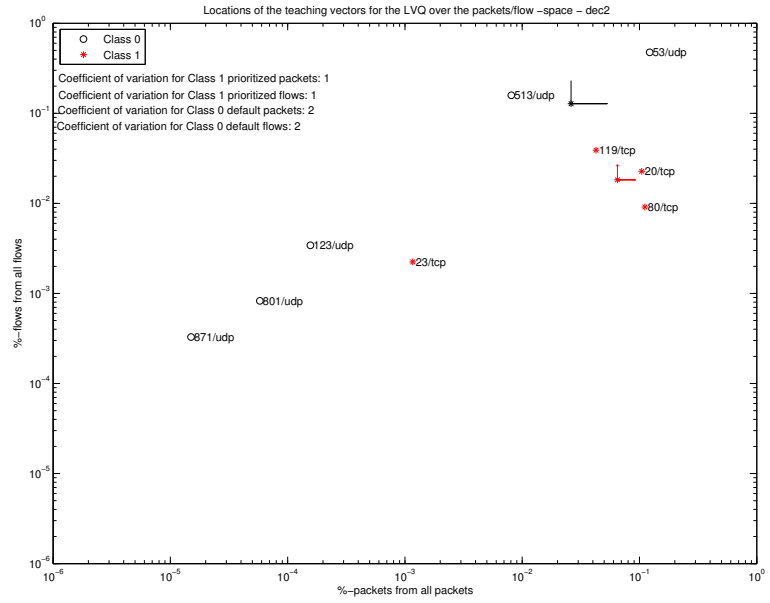
5.3 Classifier construction

We choose to apply the OLVQ1 algorithm to the problem of traffic classification since it provides an individual learning rate for each codebook vector and because it is recommended as the initial LVQ-algorithm in [5]. The algorithm itself is extensively explained and clarified in [5], so only the necessary details and the actual application to traffic classification are offered here.

As we apply the LVQ algorithm to the flow classification problem we use the basic



(a) dec1



(b) dec2

Figure 5.7: Teaching sets $\mathbb{X}$ in the log-scale for the LVQ classifier / dec1 and dec2

program package[1] explained in [78]. The process of applying the LVQ classifier to the flow classification problem is shown in Figure 5.11.

[1]Available via WWW from http://www.cis.hut.fi/nnrc/lvq_pak/

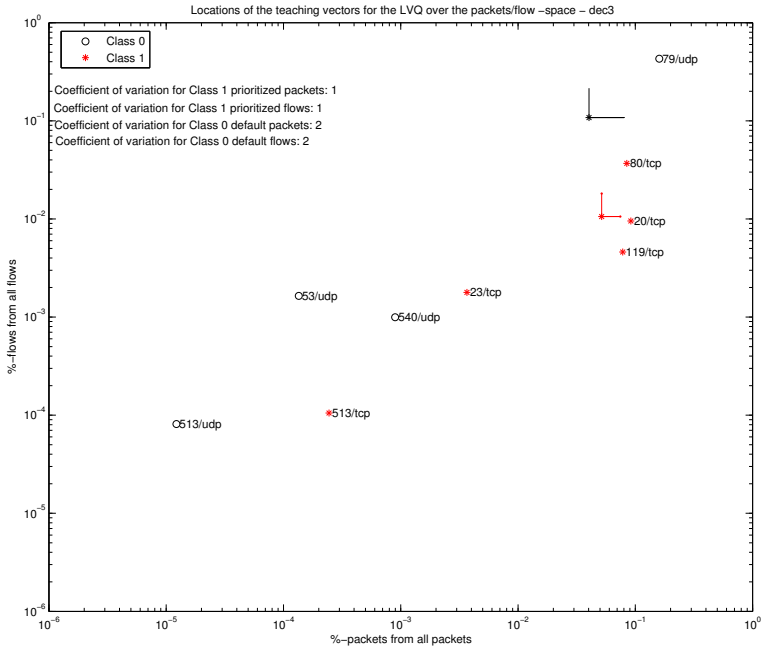


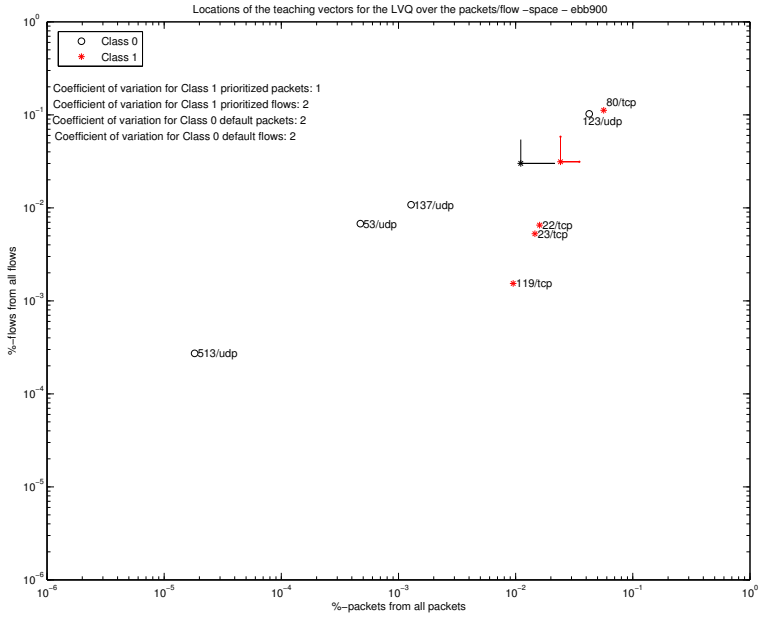
Figure 5.8: Teaching sets $\mathbb{X}$ in the log-scale for the LVQ classifier / dec3

1. We first measure traffic and perform the flow analysis in the network. After the measurement phase we have a list of all the applications seen in the network together with the total amount of packets and flows measured for these applications. We then normalize the results to the total number of packets and to the total number of flows seen in the trace. This way we make the resulting classifier less dependent on the absolute amount of packets and flows and thus more generally applicable.

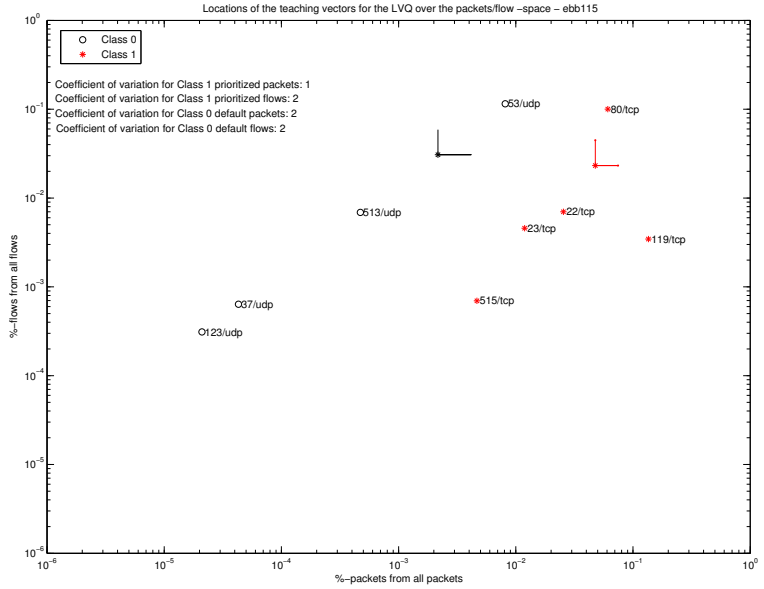
2. A set of teaching vectors, $\mathbb{X}$, is then constructed. We form a training data set of individual training samples, x, possessing clear qualifications for either prioritization, $x_{c=1}$, or default handling, $x_{c=0}$. Since the LVQ algorithm defines the borders of the packets to flows distributions special care has to be taken to select enough training data from the borderline of the two traffic types.

3. We then proceed to the initialization phase. We determine the initial codebook vectors, $v \in \mathbb{V}$, which have to be within the borders of corresponding classes by using the teaching vectors. This is done by using k-nearest-neighbor–classification where $k = 5$. At this stage we also determine the number of codebook vectors and distribute them evenly between the two classes. The OLVQ1 algorithm is then used to optimize the placing of the codebook vectors:

$$\begin{aligned} v_i[j+1] &= v_i[j] + \alpha[j](x[j] - v_i[j]) \\ &\quad \text{if } x \text{ and } v_i \text{ belong to the same class,} \\ v_i[j+1] &= v_i[j] - \alpha[j](x[j] - v_i[j]) \\ &\quad \text{if } x \text{ and } v_i \text{ belong to different classes,} \\ v_k[j+1] &= v_k[j] \text{ for } i \neq k. \end{aligned} \tag{5.2}$$



(a) ebb900



(b) ebb115

Figure 5.9: Teaching sets $\mathbb{X}$ in the log-scale for the LVQ classifier / ebb900 and ebb115

We use $\alpha_i[j]$ for every codebook vector, instead of a global $\alpha[j]$, and then determine recursively the near optimal values of $\alpha_i[j]$ for each codebook vector, v_i, individually:

$$\alpha_i[j] = \frac{\alpha_i[j-1]}{1 + s[j]\alpha_i[j-1]} \tag{5.3}$$

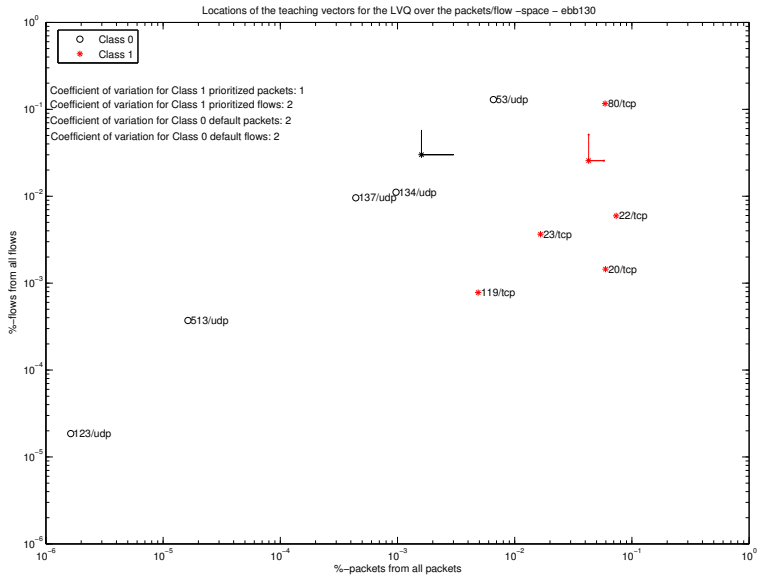


Figure 5.10: Teaching sets $\mathbb{X}$ in the log-scale for the LVQ classifier / ebb130

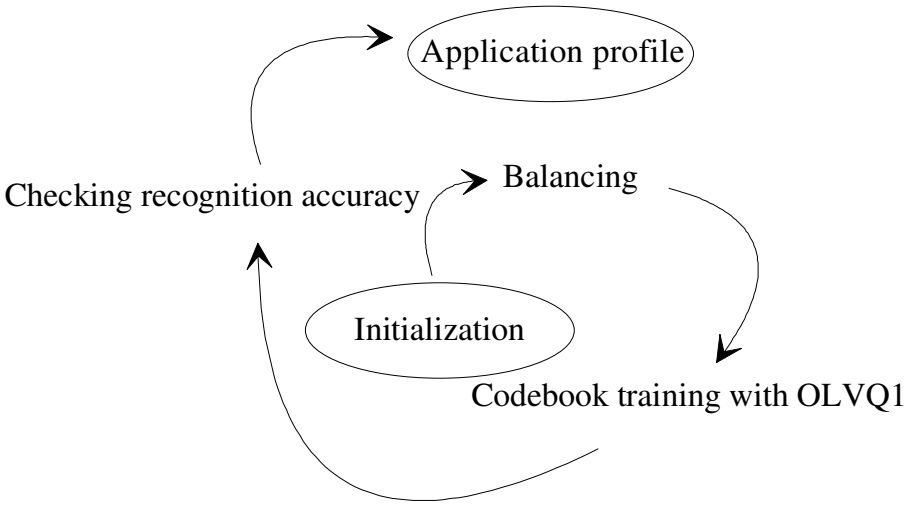


Figure 5.11: The construction of the LVQ classifier

where $s[j] = +1$ if the classification is correct (i.e., if the class of the codebook vector is the same with the teaching vector), and $s[j] = -1$ if the classification is incorrect. Here $0 < \alpha[t], \alpha_i[t] < 1$, and $\alpha[t]$ or $\alpha_i[t]$ may be constant or decrease monotonically with time. Several enhancements and modifications exist to the LVQ–algorithm [5, 78] but the algorithm presented above is used in this work.

A codebook vector, v_i, represents a reference point of the application behavior in the packet-flow –space. The actual learning process is then:

$$v_i[j+1] = (1 - s[j]\alpha_i[j])v_i[j] + s[j]\alpha_i[j]x[j] \tag{5.4}$$

The final codebook vectors, v_i, are trained using (5.4) together with the training vectors.

4. After initializing the codebook vectors we balance the amount of entries in any of the classes. This is done by adding and deleting entries according to average distances. Then one learning cycle of optimized-learning-rate LVQ (OLVQ1) is applied to the codebook vectors. In this work, the balancing is done two times for each set of codebook vectors.

5. It is suggested in [5] that the number of learning steps should be set to about 30 to 50 times the total number of codebook vectors. In this work the final codebook vectors are trained for 40 times the number of codebook vectors assigned to the classifier using (5.4) and the training vectors.

6. When the codebook vectors have been determined we can classify all of the application vectors of unknown class letting the unclassified vector be y and defining the c for v_c to be the nearest codebook vector v_i to unclassified y as:

$$c = \arg\min_i \{\|y - v_i\|\} \tag{5.5}$$

The pseudo code implementation of the traffic classification process using the LVQ algorithm is shown in Table 5.5. The flow classification and flow analysis processes are identical to those in Table 3.3 and 3.2 and therefore only the traffic classification steps are shown in Table 5.5. The whole process starts with the initialization of the codebook

Table 5.5: Pseudo code of traffic classification for 2-class LVQ classifier

process Traffic_Classification
Input $\mathbb{X}, \mathbb{P}_\mathbb{T}$;
Init $\mathbb{V}$;
  knn-classify $\mathbb{V}$ *with* $\mathbb{X}$;
Optimize placement of $\mathbb{V}$;
  $v_i[j+1] = [1 - s[j]\alpha_i[j]]v_i[j] + s[j]\alpha_i[j]x[j]$;

Flow_Analysis($\mathbb{P}_\mathbb{T}$);
  Return $\{(a, F_{norm}(a), C_{norm}(a))\}$;

Classify measurements;
Return $\mathbb{S}$;
end Traffic_Classification;

vectors $\mathbb{V}$ using the teaching samples $\mathbb{X}$. After optimizing the codebook vector placement, we do the flow analysis obtaining the relative packet and flow counts for each existing application a. The flow analysis results are then classified and these classification results determine the rule set $\mathbb{S}$. The context analysis follows using the flow classification routine outlined in Table 3.3.

5.4 Performance results of the LVQ classifier

In this section, we first observe the behavior of the 2-class LVQ classifier as seen from the network side. We then proceed to look at the selection of applications that the 2-class LVQ classifier picks up for prioritization.

5.4.1 Test environment for the simulation

The simulation methodology for the performance evaluation is as follows:

1. First we perform the flow analysis gathering the information on flows and packets on an application by running the *issim* -simulator [30]. We use 60 second flow timeouts for the flows defined by the 5-tuple $(Saddr, Daddr, Proto, Sport, Dport)$. The output consists of application and protocol identifiers with packet and flow count observed for this application.
2. With the application list (and related statistics) we form the teaching vectors manually.
3. We teach the classifier with the teaching samples and optimize the codebook vector placement according eq. 5.4. As output we get the codebook vectors that act as the actual classifier.
4. We then use the classifier to classify the measurement results to proper classes and obtain the network application profile. At this stage, we may perform the content performance analysis based on the application list.
5. Finally we use the network application profile to the traffic trace and observe the context performance in the test environment.

Finally, there exists an unrealistic aspect, however, in the test environment and methodology. When the resulting network application profile is determined and applied to a trace that we have just used for determining the network application profile, we obtain results that may be too positive, especially regarding the context performance. In real networks, we are unable to determine the network application profile beforehand. We present, however, results based on the assumption that we would have known the network application profile in advance. To compensate, we use the network application profiles from selected traces from the two network environments as the network application profiles for the rest of the traces. On these occasions we see an indication how the 2-class LVQ classifier would perform in a more realistic setting.

5.4.2 Context analysis of the 2-class LVQ classifier

In Table 5.6 we show the context performance results of our 2-class LVQ classifier in different networks with different application profiles. The results have been obtained by using a classifier constructed of the teaching samples from the network in question. After using the classifiers to classify the trace, we have obtained a suggested application profile that has then been used to determine the performance statistics. We also observe, in the second part of Table 5.6, the possibility for general applicability of a classifier as it is used in the same network environment but with different trace. In the dec-traces we have used the classifier constructed for the dec1-trace for all other dec-traces. The classifier constructed for the ebb900-trace has been used to classify the other ebb-traces. Looking at the results we see that the dec3-trace, when taught with dec3-teaching vectors, has double the flow birthrate than other dec-networks. The application factor is also very high compared to other traces. Both the outlier in the dec3-teaching vectors and the nature of the traffic in the dec3-network are the most probable cause for this. It seems

Table 5.6: Basic context performance analysis of the LVQ classifier

Performance statistics of the LVQ classifier with own application profile				
Classified network	Flow birthrate factor, B_{LVQ}	Flow space factor, S_{LVQ}	Packet classification factor, C_{LVQ}	Application factor, A_{LVQ}
dec1	23,3% at 3536s	23,7%	66,5%	0,74%
dec2	24,3% at 3564s	30,8%	70,4%	0,49%
dec3	48,6% at 3539s	47,5%	73,8%	10,23%
ebb900	52,3% at 3571s	78,9%	84,2%	0,41%
ebb115	47,4% at 3576s	75,5%	90,4%	0,54%
ebb130	64,0% at 3589s	45,3%	90,9%	0,75%
Performance statistics of the LVQ classifier with preset[a] classifier				
Classified network	Flow birthrate factor, B_{LVQ}	Flow space factor, S_{LVQ}	Packet classification factor, C_{LVQ}	Application factor, A_{LVQ}
dec2	33,0% at 3592s	37,6%	87,4%	0,71%
dec3	63,1% at 3570s	57,5%	89,3%	0,59%
ebb115	26,4% at 3664s	37,1%	30,0%	0,43%
ebb130	21,6% at 3571s	16,6%	37,6%	0,41%

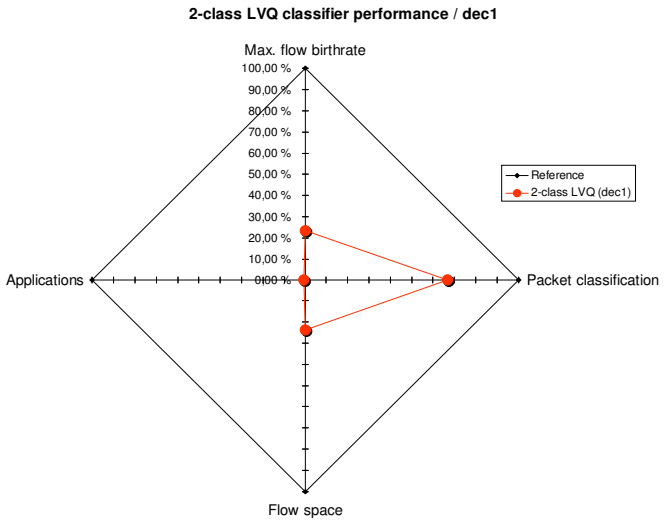
[a]dec2 and dec3 classified with dec1-classifier, and ebb115 and ebb130 classified with ebb900-classifier.

that the dec3-network is much more active than the other dec-networks. This statement is backed up by looking at the reference point results in Table 4.4 where the flow birthrate factor is higher than in other networks.

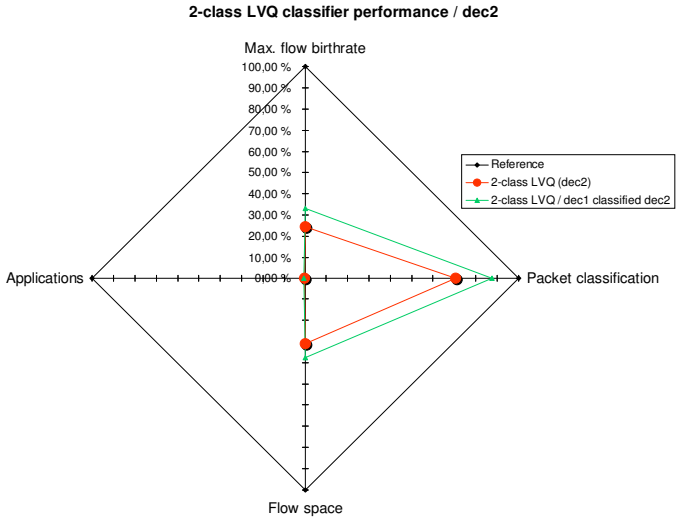
The use of flow space is quite high but seems to depend up on the network and traffic characteristics. The packet classification factor is also quite high, except for the cases in ebb115 and ebb130 when the classifier is taught with ebb900. It should be noted that the high packet classification factors are obtained with the very minimum amount of applications, less than one percent in all cases except the aforementioned dec3.

In Figures 5.12 to 5.15 we show all the performance data in the quadrilateral model for each network in the three previous cases. Figures 5.12 to 5.15 show that the LVQ-classifiers perform in equal manners in the same network environment. The greatest variations seem to be with the packet classification factor.

Summarizing Figures 5.12 to 5.15 we see that in all of the network environments our 2-class LVQ classifier is able to segregate a small set of applications (less than 1%) that still carry a substantial amount of packets, starting from classifying over 30% of all packets to the priority and rising to classify as high as over 90% of all packets to priority class. The dec3-performance with its own classifier should be regarded as a special case when the choice of teaching vectors combined with slightly more active traffic behavior, has influenced the classifier performance and resulted in approximately 10% of the applications to be classified and causing two times higher flow birthrate factor



(a) dec1



(b) dec2

Figure 5.12: Performance of the 2 class LVQ classifier / dec1 and dec2

and one and a half times higher use of flow space than with other dec-traces.

It seems that the LVQ -classifier produces reasonable performance figures in a particular network environment provided that the teaching vectors have been carefully selected. In conclusion, the observation of our basic 2-class LVQ classifier indicates that the use of the 2-class LVQ classifier is justifiable and does not result in excessive added performance requirements although the 2-class LVQ classifier might not be the best choice for optimizing context performance.

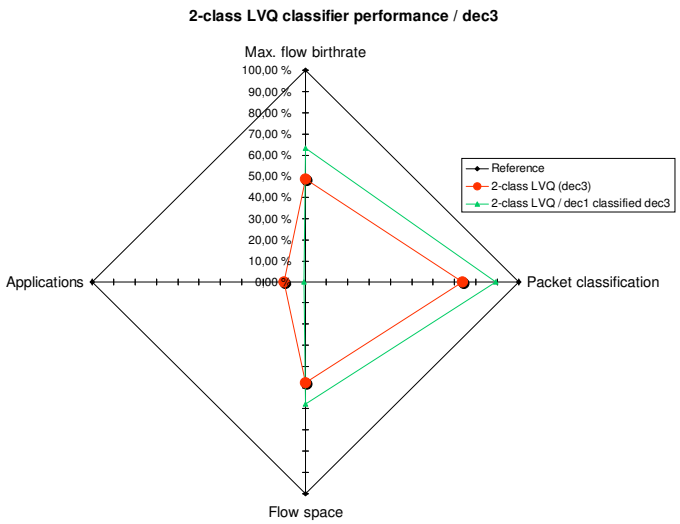


Figure 5.13: Performance of the 2 class LVQ classifier / dec3

5.4.3 Content analysis of the 2-class LVQ-classifier

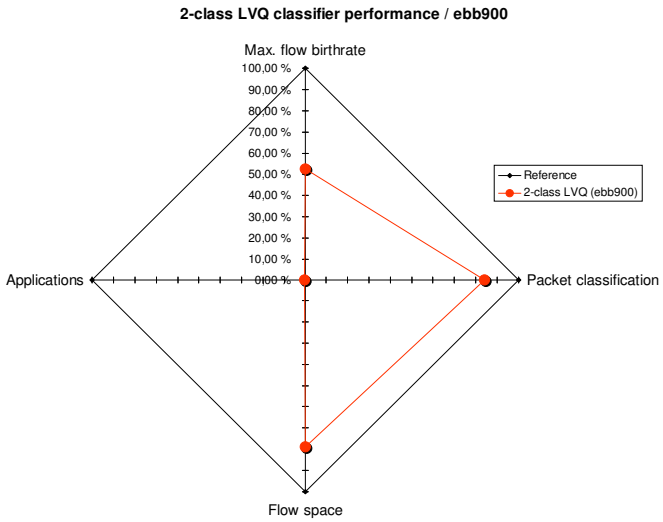
The content analysis statistics are shown in Tables 5.7 and 5.8. Looking at the coefficient of variation we can see that in all traces, except dec3 with its own codebook as classifier, the Class 1 coefficient of variation differs in the order an magnitude from that of the Class 0. The results seen for Class 1 resemble the results of the priority class with static application classifier and the threshold classifier with 2% threshold.

In dec3, the sloppy choosing of teaching vectors results in a behavior similar to that of the threshold classifier with 10% threshold. However, when dec3 is classified with dec1-codebook the content analysis statistics change back to the level found in other LVQ-classified cases.

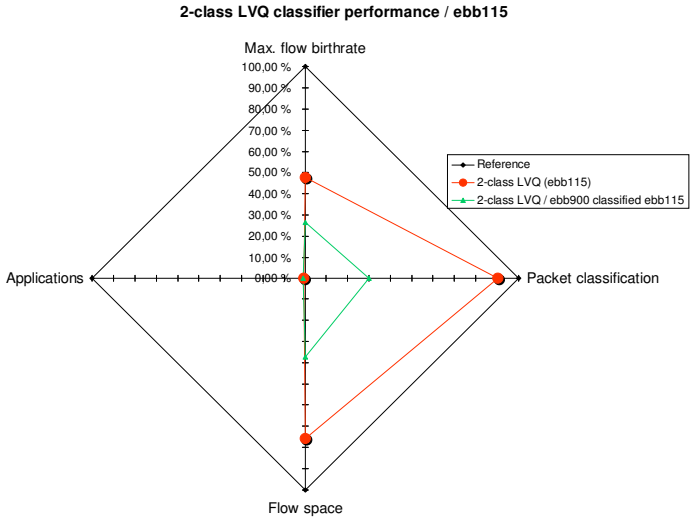
The locations of the priority applications in the dec-traces in the packet-flow –space are shown in Figures 5.16 to 5.18. We can see that the priority applications are located in the high relative packet share area. When comparing the average class locations and standard deviation based application range, we can see that Class 1 is much more restricted in behavior to the point that the range vectors for Class 1 are hardly visible in Figures 5.16 to 5.18. These observations on the Class 1 applications are in accordance with the locations of the teaching vectors for these traces.

An interesting phenomenon may be observed when looking at dec3-trace in Figure 5.18. When classified with its own codebook, Class 1 area is not a continuous one, but comprised of two areas. Although, when observing the content statistics, it seems that in this case this type of behavior is a result of poor choice of teaching samples, it indicates that the 2–class LVQ–classifier has the capability to choose very complex sets of classified applications. The key factor in achieving this is to choose the teaching vectors carefully.

The locations of the priority applications in the ebb-traces in the packet-flow –space are shown in Figures 5.19 to 5.21. We can see that the priority applications share a similar behavior to those found in dec-traces. Class 1 behavior is restricted to the high relative



(a) ebb900



(b) ebb115

Figure 5.14: Performance of the 2 class LVQ classifier / ebb900 and ebb115

packet share section of packet-flow –space. Also with the ebb-traces the range vectors for Class 1 are hardly visible in Figures 5.19 to 5.21.

Table 5.9 shows some of the applications (well-known port number area and some selected samples in the rest of the port number area) in the dec-traces that the 2-class LVQ-classifier has classified to a higher priority. Table 5.9 shows that although different classifiers result in slightly different network application profiles, the core applications are quite the same within a network environment during a relatively short time period. Outside the teaching samples (refer to Table 5.4) we note that electronic mail -services (port 25/tcp) and some telnet -like services (port 514/tcp) are assigned priority classification and most notably and consistently the alternative www-application (port 8080/tcp) is prioritized in all of the traces. The dec3-trace results show that with the combination of slightly careless selection of teaching vectors and unexpected network behavior, the

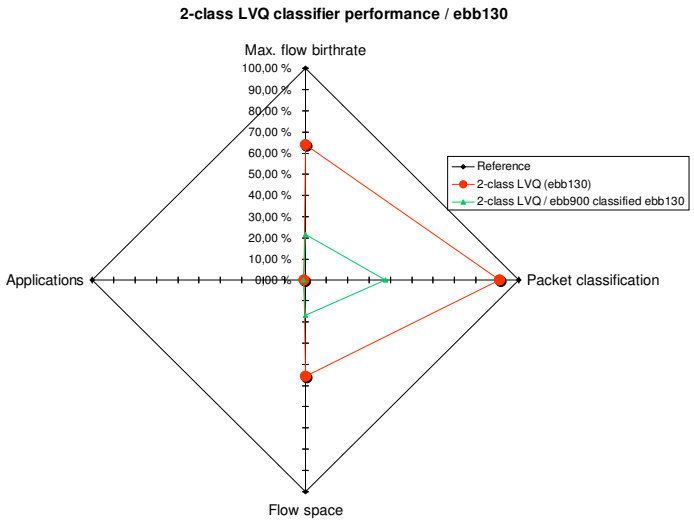


Figure 5.15: Performance of the 2 class LVQ classifier / ebb130

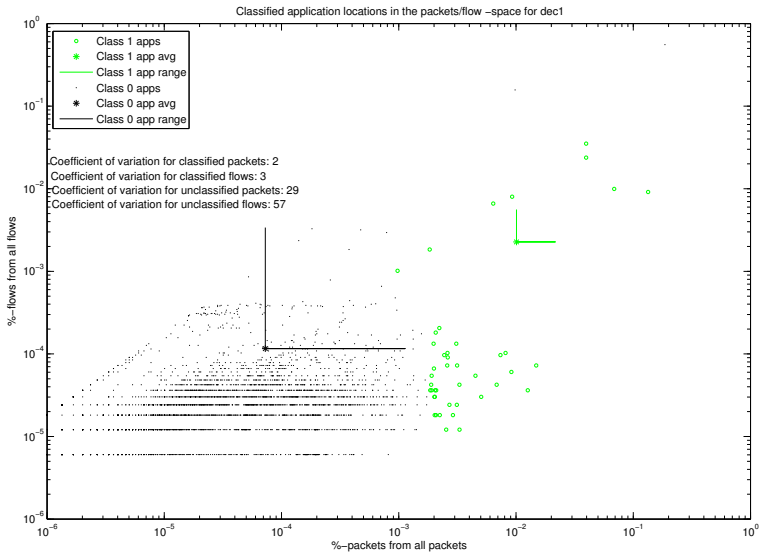


Figure 5.16: Locations of priority applications using the LVQ-classifier / dec1

number of prioritized applications rises to a very high level making it difficult for the network managers to pick out the final priority set.

Table 5.10 shows some of the applications (in the well-known port number area and some selected samples in the rest of the port number area) that the 2-class LVQ-classifier has given priority classification in the ebb-traces. Table 5.10 shows that a similar behavior regarding the application diversity continues in the ebb-traces as it did with the dec-traces. Different classifiers result in slightly different network application profiles but the core of the priority classified applications is similar within a network environment. Furthermore, this consistent behavior is apparent even when using a classifier constructed based on previous measurements. Finally, as with the dec-traces, one must take note that

Table 5.7: Content statistics for dec-traces with 2-class LVQ classification

Content statistics for dec-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
dec1	Class 1 pkt	1,0087%	0,1485%	0,0977%	13,3956%	2
dec1	Class 1 flw	0,2271%	0,042%	0,0012%	3,5157%	3
dec1	Class 0 pkt	0,0073%	0,1697%	0,00003%	18,6375%	29
dec1	Class 0 flw	0,0116%	0,5244%	0,0006%	55,5973%	57
dec2	Class 1 pkt	1,8470%	0,1576%	0,1165%	11,1124%	2
dec2	Class 1 flw	0,4479%	0,051%	0,0025%	3,8977%	2
dec2	Class 0 pkt	0,0070%	0,113002%	0,00003%	12,2226%	21
dec2	Class 0 flw	0,0118%	0,4557%	0,0006%	47,5911%	49
dec2 / dec1 cbk[a]	Class 1 pkt	1,2889%	0,158645%	0,1165%	11,1124%	2
dec2 / dec1 cbk	Class 1 flw	0,3427%	0,055351%	0,00123%	3,8977%	3
dec2 / dec1 cbk	Class 0 pkt	0,00634%	0,11157%	0,00003%	12,2226%	22
dec2 / dec1 cbk	Class 0 flw	0,0115%	0,4552%	0,0006%	47,5911%	50
dec3	Class 1 pkt	0,0938%	0,1396%	0,01197%	9,1704%	6
dec3	Class 1 flw	0,0437%	0,0911%	0,000300%	7,7441%	9
dec3	Class 0 pkt	0,0034%	0,1457%	0,00002%	16,0185%	44
dec3	Class 0 flw	0,00623%	0,3898%	0,0003%	42,9367%	64
dec3 / dec1 cbk	Class 1 pkt	1,2363%	0,1392%	0,1090%	9,1704%	2
dec3 / dec1 cbk	Class 1 flw	0,5412%	0,083%	0,0018%	7,7441%	3
dec3 / dec1 cbk	Class 0 pkt	0,0048%	0,1460%	0,00002%	16,0185%	31
dec3 / dec1 cbk	Class 0 flw	0,00674%	0,3915%	0,0003%	42,9367%	58

[a]cbk stands for codebook

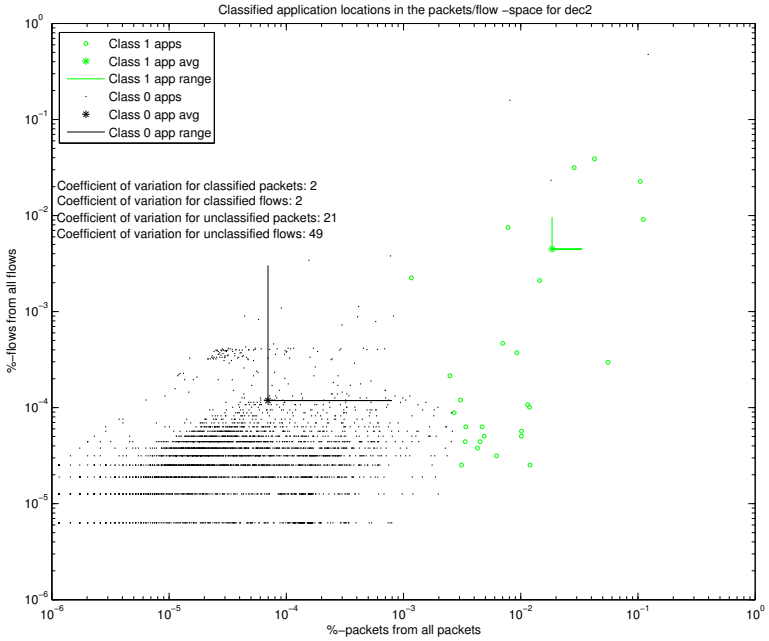
although the application profile seems to be relatively stable the context performance may still change. The applications prioritized in Table 5.10 are also as interactive as possible among the applications existing in traces. Outside the teaching samples (refer to Table 5.4) we note that in some cases the electronic mail -services (port 25) and some ftp-service (port 20) are given priority and most notably and consistently in the ebb-traces the Xwin-service (port 6000) is prioritized in all of the traces.

When comparing the core applications prioritized in Tables 5.9 and 5.10 we see a slight change in the application profiles. The use of mail-services has expanded over several ports (25,110). The need for secure data exchange in the network has resulted in applications like ssh and https (ports 22 and 443). The total amount of applications, however,

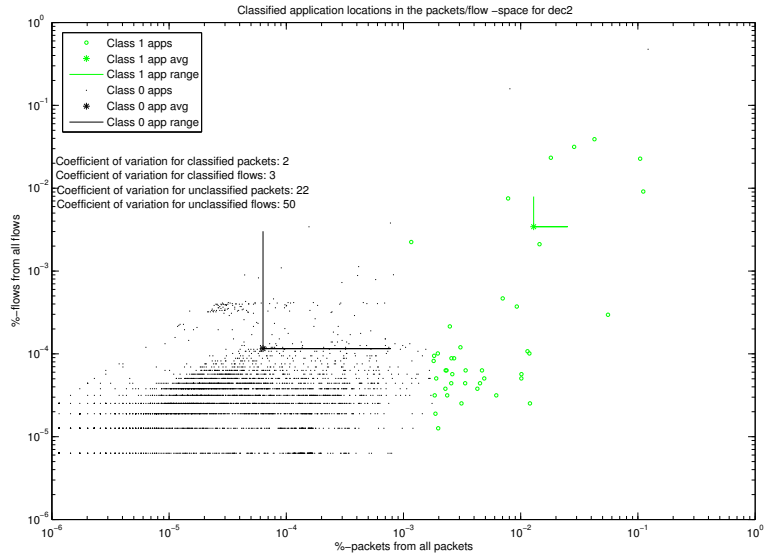
Table 5.8: Content statistics for ebb-traces with 2-class LVQ classification

Content statistics for ebb-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
ebb900	Class 1 pkt	2,9840%	0,242%	0,4875%	18,1552%	1
ebb900	Class 1 flw	1,0144%	0,1347%	0,0231%	11,1828%	2
ebb900	Class 0 pkt	0,00277%	0,0470%	0,00009%	4,2458%	17
ebb900	Class 0 flw	0,0081%	0,1234%	0,0014%	10,2406%	15
ebb115	Class 1 pkt	2,4900%	0,2249%	0,2568%	13,5456%	1
ebb115	Class 1 flw	0,7678%	0,1224%	0,0044%	10,0193%	3
ebb115	Class 0 pkt	0,0024%	0,0147%	0,0001%	0,8127%	6
ebb115	Class 0 flw	0,0088%	0,1362%	0,0015%	11,5094%	16
ebb115 / ebb900 cbk	Class 1 pkt	3,0873%	0,2247%	0,4885%	13,5456%	1
ebb115 / ebb900 cbk	Class 1 flw	0,9499%	0,1224%	0,0044%	10,0193%	2
ebb115 / ebb900 cbk	Class 0 pkt	0,0027%	0,0178%	0,0001%	0,8127%	7
ebb115 / ebb900 cbk	Class 0 flw	0,0089%	0,1362%	0,0015%	11,5094%	16
ebb130	Class 1 pkt	2,4894%	0,257%	0,2552%	16,4349%	1
ebb130	Class 1 flw	0,6057%	0,128%	0,0074%	11,6482%	3
ebb130	Class 0 pkt	0,0023%	0,0122%	0,00008%	0,6585%	6
ebb130	Class 0 flw	0,0105%	0,157%	0,0019%	12,9275%	16
ebb130 / ebb900 cbk	Class 1 pkt	3,2932%	0,257%	0,4878%	16,4349%	1
ebb130 / ebb900 cbk	Class 1 flw	0,8177%	0,1279%	0,0074%	11,6482%	3
ebb130 / ebb900 cbk	Class 0 pkt	0,0027%	0,016%	0,00008%	0,6585%	7
ebb130 / ebb900 cbk	Class 0 flw	0,0105%	0,1573%	0,0019%	12,9275%	16

has remained relatively constant, between 7000 and 12000 applications seen in any of the traces. Also the 2-class LVQ classifier detects quite a constant, and a low, amount of priority applications, ranging from as low as 24 applications up to 651 applications. Based on this, it would be safe to say that, although the number of applications used in a network, or prioritized with a 2-class LVQ classifier, remains relatively constant, the distribution and locations of these applications in the port-space varies. This variation is significant and disregarding it, and using constant application sets, would result in inconsistent network application profiles.



(a) dec2

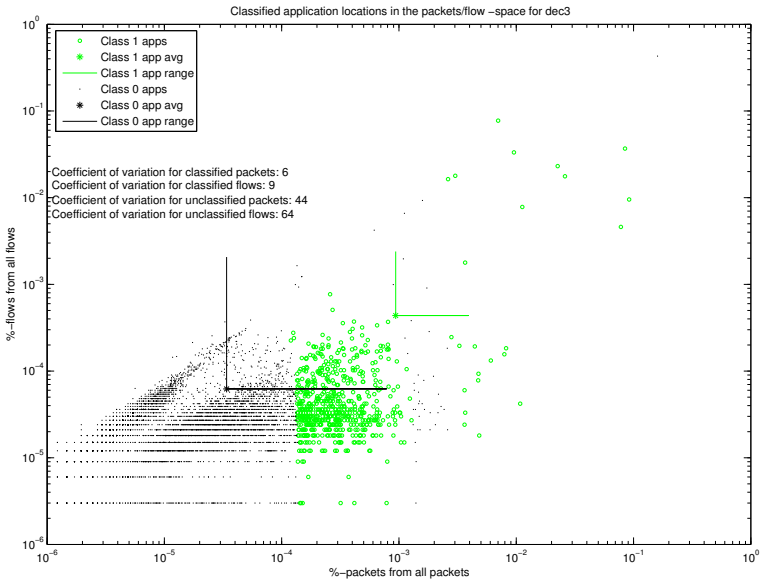


(b) dec2 with dec1 codebook

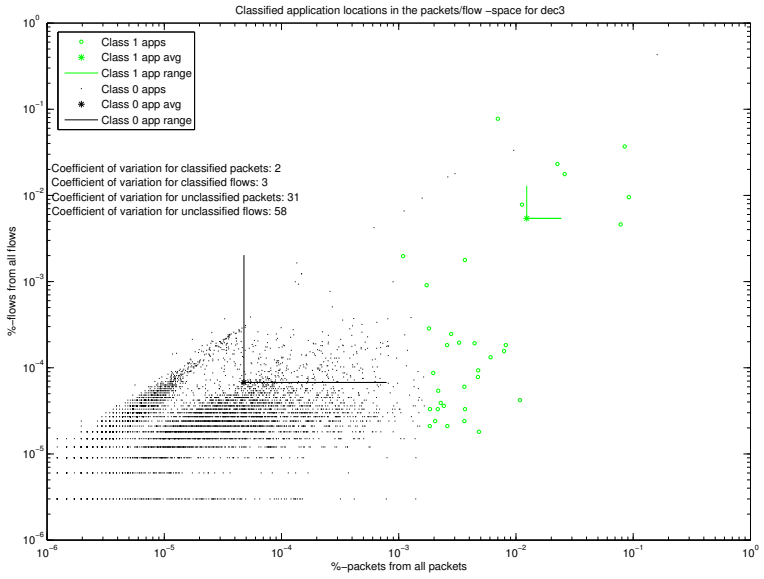
Figure 5.17: Locations of priority applications using the LVQ-classifier / dec2

5.5 Conclusions

In this chapter we observed the accumulated statistics of flows and packets on an application and based the selection of the application profile on these statistics. The measurements gather accumulated packets/flow per application data and with the application oriented traffic classification scheme we try to differentiate the important applications



(a) dec3



(b) dec3 with dec1 codebook

Figure 5.18: Locations of priority applications using the LVQ-classifier / dec3

from less important. We use the LVQ algorithm to determine the areas of important traffic in the packet-flow –space. The importance of different applications is determined by the choice of teaching vectors and, in this work, we concentrate on trying to find applications that produce a lot of packets in few flows.

Figure 5.22 shows the method how the LVQ classifier is placed in a measurement based traffic classification environment. Figure 5.22 also indicates the points of what we can vary in the classifier construction process:

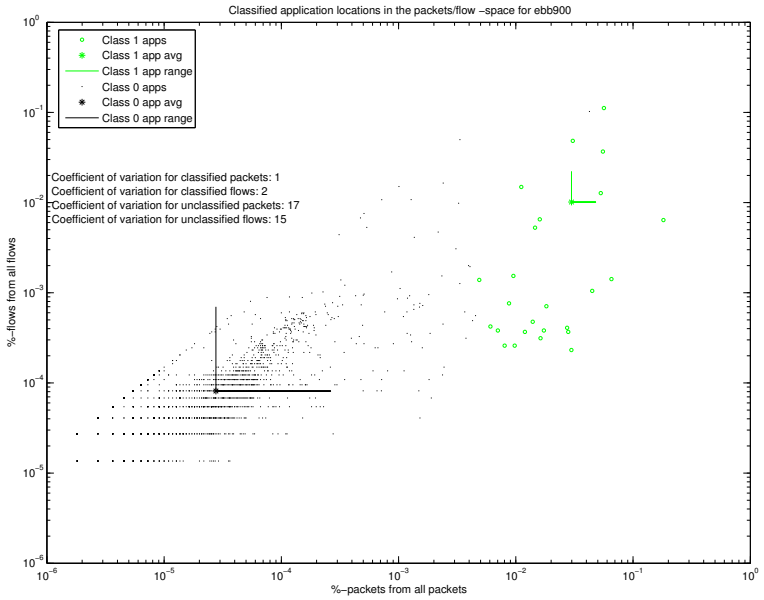
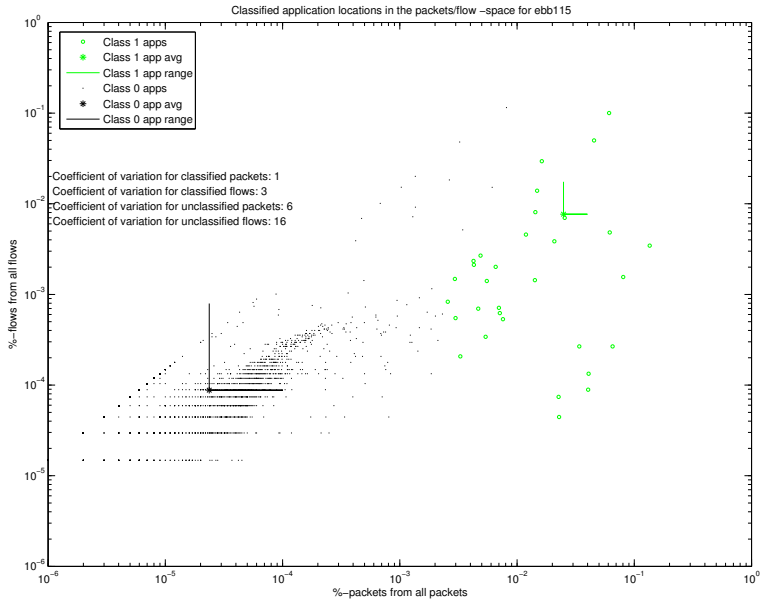


Figure 5.19: Locations of priority applications using the LVQ-classifier / ebb900

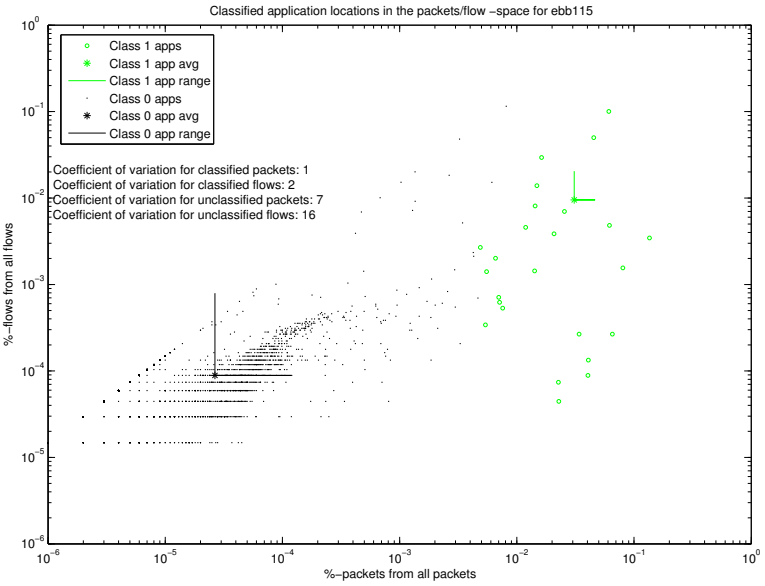
Table 5.9: Prioritized application profiles obtained with LVQ classifiers / dec-traces

Application lists		
Classified network	Individual network application profile	Profile obtained using the preset (dec1) classifier
dec1	**20/tcp**[a], **21/tcp**, **23/tcp**, 25/tcp, **80/tcp**, **119/tcp**, 520/udp, 540/tcp, **1307/udp**, **8080/tcp (10 out of total 43 applications (4 udp, 39 tcp))**	
dec2	20/tcp, 21/tcp, 23/tcp, 80/tcp, 119/tcp, 514/tcp, 520/udp, 8080/tcp **(8 out of total 26 applications (4 udp 22 tcp)).**	25/tcp **(1 shown of 15 added (1 udp 14 tcp)).**
dec3	0/udp, 1/udp, 20/tcp, 21/tcp, 23/tcp, 25/udp, 25/tcp, 28/udp, 70/tcp, 80/udp, 80/tcp, 119/tcp, 123/udp, 137/udp, 513/tcp, 514/udp, 514/tcp, 6000/tcp, 6969/tcp, 8000/tcp, 8001/tcp, 8080/udp, 8080/tcp **(23 out of total 651 applications (45 udp 606 tcp)).**	20/tcp, 21/tcp, 23/tcp, 25/tcp, 53/tcp, 80/tcp, 119/tcp, 123/udp, 520/udp, 8080/tcp **(total of 34 (617 less) applications (12 udp 22 tcp).**

[a]Boldfaced application identifiers indicate applications prioritized in all dec-traces.



(a) ebb115

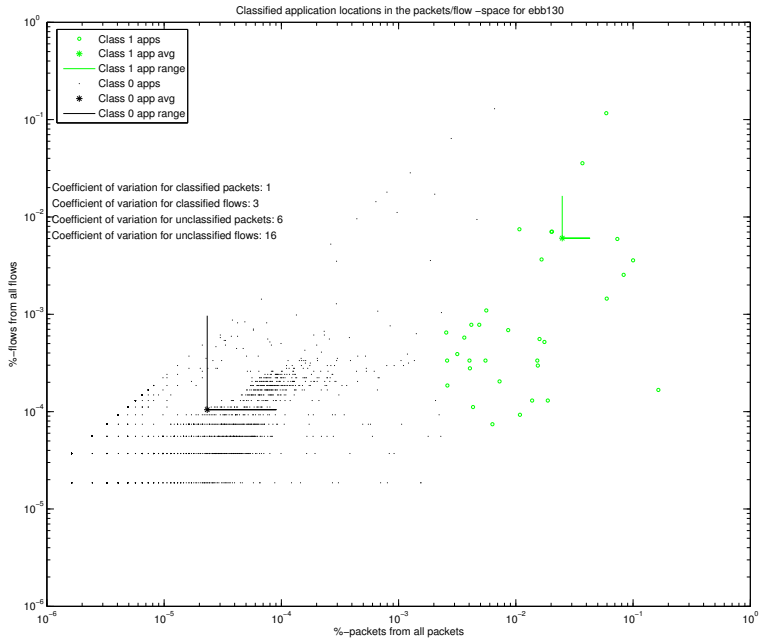


(b) ebb115 with ebb900 codebook

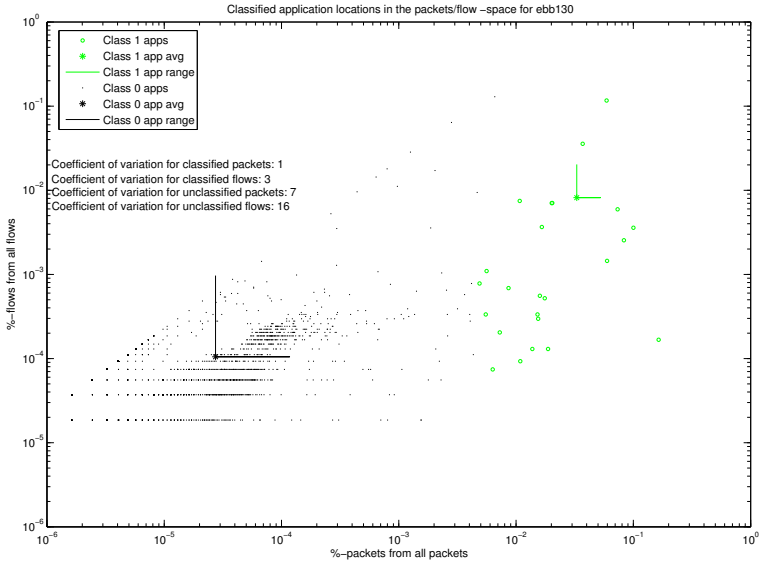
Figure 5.20: Locations of priority applications using the LVQ-classifier / ebb115

- The definition of flow; the level of flow granularity.
- The LVQ parameters; number of codebook vectors, number of learning steps and the learning rate.
- The process of constructing teaching vectors from the measurement results.

Results show that our 2-class LVQ classifier is successful in detecting applications from



(a) ebb130



(b) ebb130 with ebb900 codebook

Figure 5.21: Locations of priority applications using the LVQ-classifier / ebb130

the network as instructed by the teaching vectors. We emphasize that this detection is based on teaching samples and by using different kinds of teaching samples the resulting network application profile could be notably altered. The selection of teaching samples is based on the subjective value (in this work, the author's) of particular applications. We note that, albeit some of the applications have faded away and others taken their place in the networks observed (dec and ebb), no substantial change in the overall behavior in

Table 5.10: Prioritized application profiles obtained with LVQ classifiers / ebb-traces

Application lists		
Classified network	Individual network application profile	Profile obtained using the preset (ebb900) classifier
ebb900	**0/udp**[a], **22/tcp**, **23/tcp**, **53/tcp**, **79/tcp**, **80/tcp**, **119/tcp**, **801/udp**, **1021/tcp**, **6000/tcp (10 out of total 25 applications (3 udp, 22 tcp))**	
ebb115	0/udp, 20/tcp, 22/tcp, 23/tcp, 25/tcp, 52/udp, 53/tcp, 79/tcp, 80/tcp, 119/tcp, 139/tcp, 515/tcp, 719/udp, 794/tcp, 796/tcp, 801/udp, 973/udp, 6000/tcp **(18 out of total 32 applications (4 udp 28 tcp)).**	0/udp, 20/tcp, 22/tcp, 23/tcp, 25/tcp, 53/tcp, 79/tcp, 80/tcp, 119/tcp, 139/tcp, 794/tcp, 796/tcp, 801/udp, 6000/tcp **(18 out of total 25 applications (2 udp 23 tcp).**
ebb130	0/udp, 20/tcp, 22/tcp, 23/tcp, 53/tcp, 79/tcp, 80/tcp, 119/tcp, 139/tcp, 801/udp, 6000/tcp **(11 out of total 33 applications (2 udp, 31 tcp)).**	0/udp, 20/tcp, 22/tcp, 23/tcp, 53/tcp, 79/tcp, 80/tcp, 119/tcp, 801/udp, 6000/tcp **(10 out of total 24 applications (2 udp 22 tcp)).**

[a]Boldfaced application identifiers indicate applications prioritized in all ebb-traces.

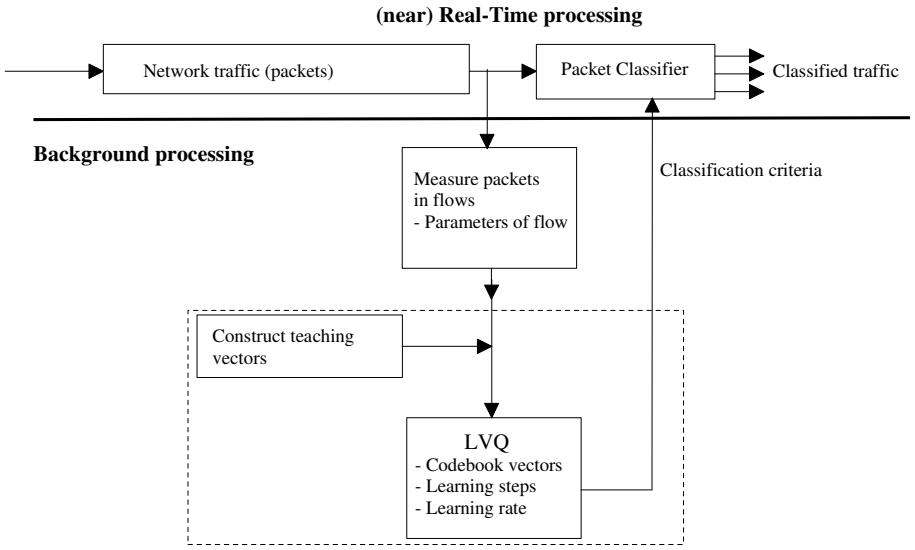


Figure 5.22: LVQ classifier in a measurement based traffic classification environment

the packet-flow –space patterns has occurred. Therefore, each network seems to have its own application profile that our 2-class LVQ classifiers, based on traffic pattern analysis, seem to be quite able to extract.

Within the measurements lies also the weak side of our 2-class LVQ classifier. The application profile is determined via measurements and therefore portrays a picture of the past. There is no guarantee that the port numbers determined by the classifier are in active use, even in the near future. By keeping the time between measuring and determining the classifier and the actual deployment of the classifier as short as possible, we might be able to avoid some of the negative effects. Using the 2-class LVQ classifier, or any predictive measurement based system, however, we risk the possibility of being unable to foresee the future.

Looking at the context performance of the 2-class LVQ classifier and combining the observations with those on the application lists we find that most of the time the 2-class LVQ classifier is able to get the best of both worlds: Relatively controlled use of resources is combined with a good effort on bringing a limited and contained set of applications to priority. Other, previously introduced, classifiers like the threshold-classifier and the packet count classifiers have not been able to do this. However, we note that the reasonable use of resources is achieved with careful selection of the teaching samples. We stated the goals of traffic classification in Table 5.3 aiming to combine the long held traffic flows and the externally controlled traffic flows into one class and the short duration transaction traffic to another. Cross-referencing to Tables 5.9 and 5.10 we see that we have succeeded in dividing the traffic into two classes that contain applications that behave in different ways in the packet-flow –space. Those applications that we have chosen to prioritize look like ones that produce relatively long held traffic flows.

The threshold classifier and the static application –classifier compete well with the 2-class LVQ classifier in performance. In fact, looking from the performance point of view, the threshold(2% and 10%)-classifier and the static applications –classifiers perform, in some cases, as well or even better than the 2-class LVQ classifier. The differences have to be searched from the examination of the application profiles and the mechanisms used to update the network application profile. Based on previous analysis on the static applications –classifier, we observe that the performance of the static application classifier is in its own class, but the static application set is unable to provide the customer the best possible priority application profile at all times. With the threshold-classifier the contents of the application lists are dynamic, because applications may enter and exit according to their usage. The weakness lies in the fact that the rigidity of the threshold may in some cases result in important applications to be left out of the priority list. Therefore, if looking at the classifiers purely from the network optimization point of view, the 2-class LVQ classifier does not outperform other classifiers. On the contrary, with sloppy construction and poor choice of teaching samples for the classifier, the dynamic nature and the application oriented creation of classification criteria give the 2-class LVQ classifier an unstable nature that has to be carefully supervised.

Having analyzed the strong and weak points of the 2-class LVQ classifier, it seems that by using a carefully designed classifier we might be able to improve the context performance of a packet forwarding system while offering the user a rational and restricted application profile to be prioritized. The 2-class LVQ classifier picks up applications in a flexible manner taking into account the changes in application usage and presence. The downside is that the 2-class LVQ classifier, or any dynamic classifier for that matter, has difficulties in guessing beforehand how much traffic will be classified and thus creates problems in network capacity allocation.

We have now established a method to divide the traffic into two classes. As this initial

division to two classes has provided promising results we will now continue to analyze the traffic with varying flow timeout values. We will extend the method of 2-class LVQ classifier to enable us to perform multi-class classification via consecutive classifications of measurement results with different flow timeout values.

Chapter 6

Multi-class traffic classification using flow analysis and the LVQ algorithm

This chapter introduces the extension on the use of the 2-class LVQ classifier concept to the task of classifying traffic to multiple classes. In Chapter 3 we discussed how the flow count gives an indication how bursty and fragmented the sending of the packets by the application is, and in the beginning of the previous chapter we observed this behavior with a constant timeout. Chapter 3 saw also the discussion on changing the timeout value and observing the flow count to reveal more application characteristics. Next we will proceed to develop the analysis process using the flow count values measured when changing the flow timeout value.

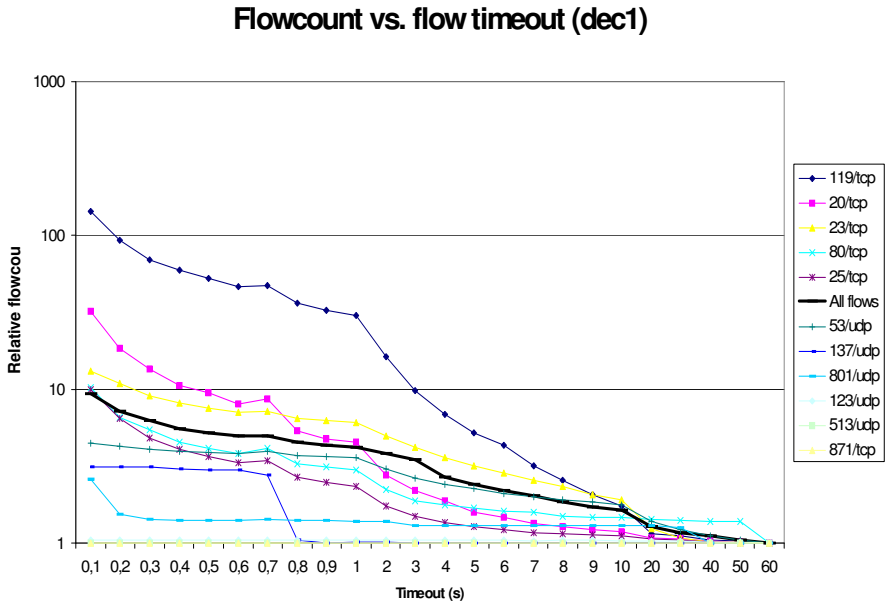
6.1 Flow analysis for multi-class classification

Running the flow analysis with different timeout values we compare the flow count values of the applications to the 60 second value. In Figures 6.1 to 6.5 we observe the flow count of selected applications as a function of the timeout. We also show how the total number of flows changes when all the applications are given priority. Looking at Figures 6.1 to 6.5 we see an anomaly where the flow count decreases at the same time the flow timeout is decreasing. The phenomenon can not be explained and is suggested to be a hard-to-find programming error in our flow analysis -software. However, the trend of the flow count behavior is evident from the figures.

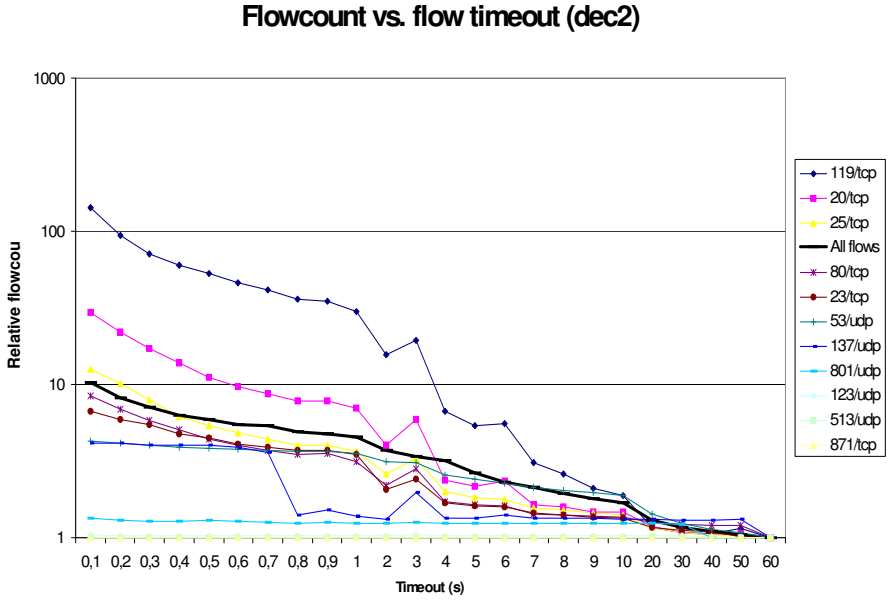
Looking at Figures 6.1 to 6.5 we can make some distinct observations:

- TCP and UDP -based applications seem to behave differently. The flow count of the TCP -based applications seems to be increasing as the timeout decreases whereas the flowcount with UDP -based applications remains practically constant.

- Telnet and ssh act in a very similar way (in networks where found together). These are protocols that at the time of measurements had very similar uses.



(a) dec1



(b) dec2

Figure 6.1: Flow count vs. the changing flow timeout in dec1 and dec2

- There seems to be a clear separation of ports 22, 23, and 119 (ssh, telnet, and nntp) from ports 25 (smtp) and port 80 (http).
- Port 80, the web-traffic, is found very near the udp-traffic thus indicating the similarity of these traffic types as far as flow count is concerned. This is somewhat expected since the www-traffic consists of relatively short bursts of information sent in the 'slow start' -phase of a TCP connection. These bursts of connections may very well resemble the flow traffic characteristics of some udp-traffic.

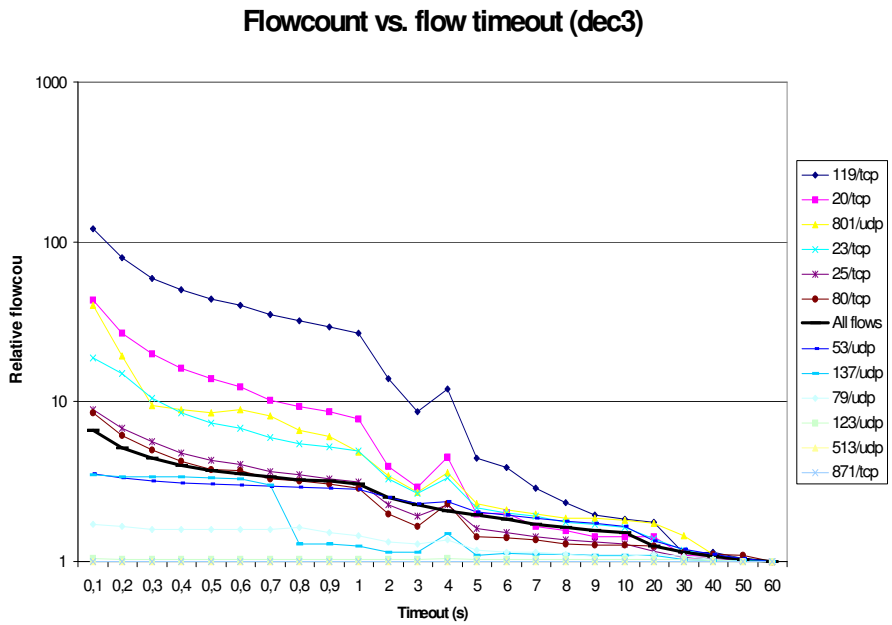


Figure 6.2: Flow count vs. the changing flow timeout in dec3

- In the tct-trace the two interactive services (real-audio in port 5900 and IP phone in port 22555) behave in distinctively different, step-wise, manner than any of the other applications. It would also appear that the flow count of these interactive applications grows very high as the flow timeout decreases.

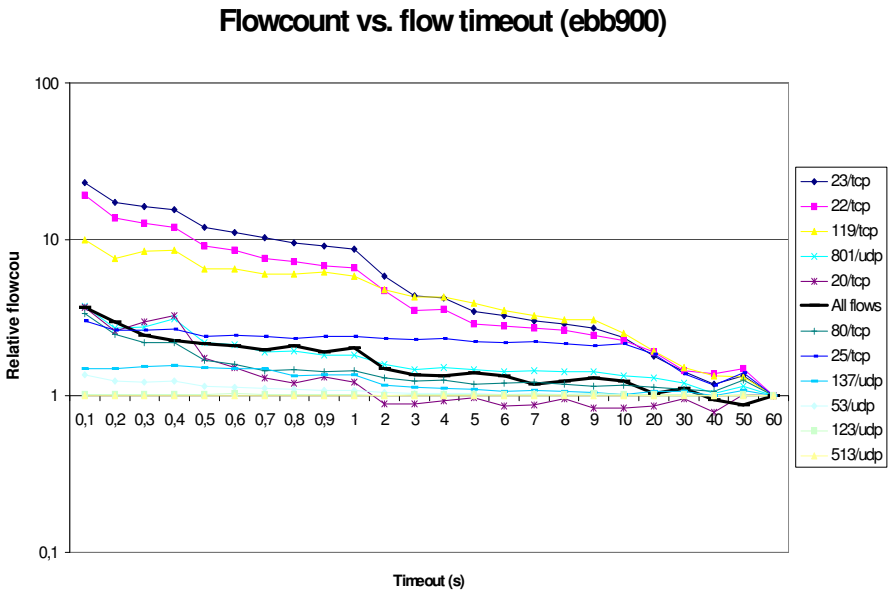
Applications of different nature (real-time, elastic, rigid) seem to behave in different manners when observing the flow count with increasing flow timeout values. This result is further clarified in Figure 6.6.

Furthermore, as previously indicated in [41], since the volume and duration of flows are correlated to higher level protocol, it would appear that traffic and application types could be differentiated into two or more classes based on observations of applications' respective flow counts with varying timeout.

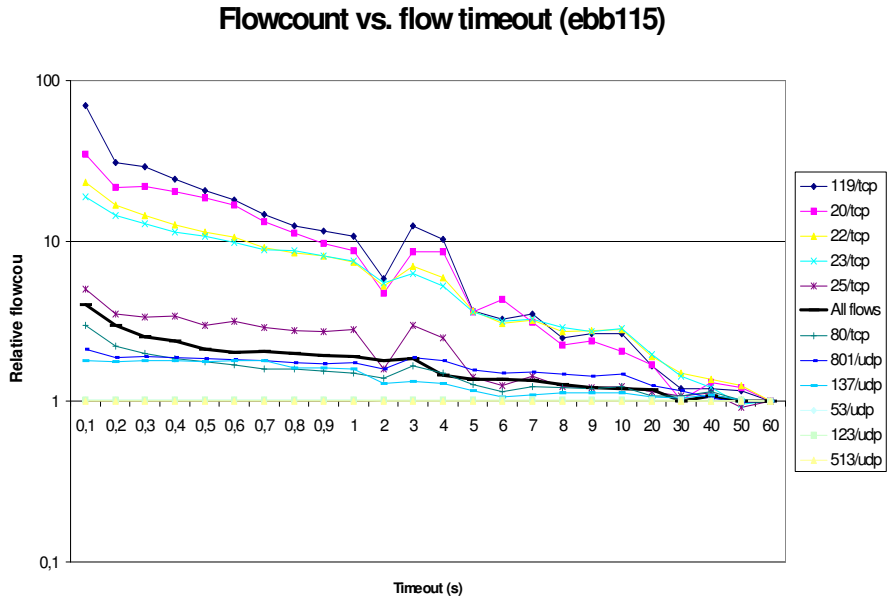
At this point we introduce an extension to the flow analysis presented in the previous chapter. Looking at figures 6.1 and 6.3 we see that the flow count changes in a different manner with different applications. Since the objective of the policy rules, in this work, is to choose applications that send a lot of packets in long bursts, we will aim to limit out any applications that either do not have high enough packet count or have a too high flow count at short flow timeouts. Therefore, we will observe the changes in packet and flow count values as the flow timeout decreases. Since the packet count for an application stays constant the major emphasis will be on observing the change in flow count of an application with different flow timeout values.

We will utilize the flow timeout values of $\tau_1 = 0.1s$, $\tau_2 = 1s$ and $\tau_3 = 10s$. This division and the values of timeouts are somewhat arbitrarily chosen based on figures 6.1 and 6.3 and might be altered depending on the network where this concept would be applied.

In the extended form of flow analysis for the multi-class LVQ traffic classifier the following



(a) ebb900



(b) ebb115

Figure 6.3: Flow count vs. the changing flow timeout in ebb900 and ebb115

statistics are gathered:

1. The application identifier a.
2. For each a with different flow timeout value τ we measure the number of flows and determine $F_{norm}(a)$ according to eq. 3.10.
3. For each a with different flow timeout value τ we measure the number of packets

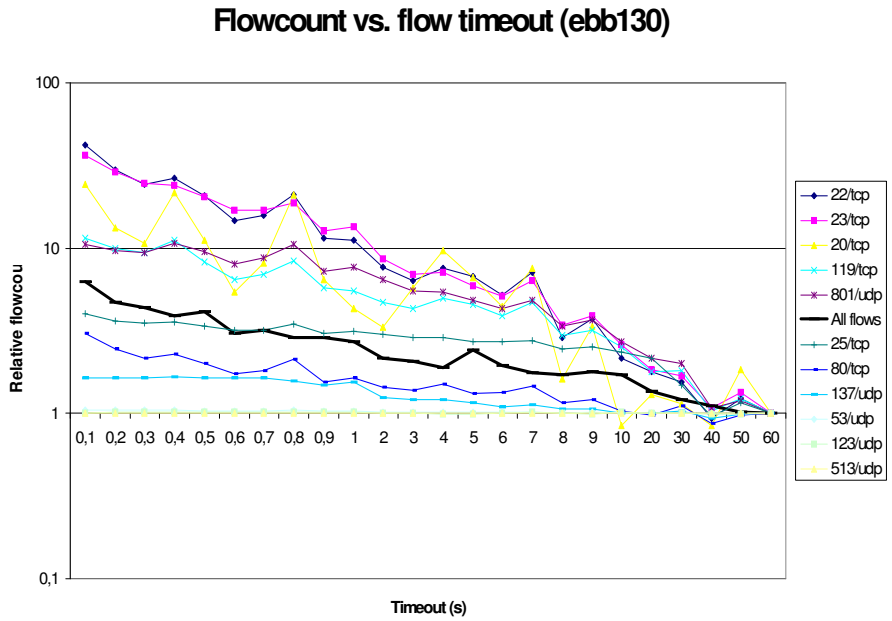


Figure 6.4: Flow count vs. the changing flow timeout in ebb130

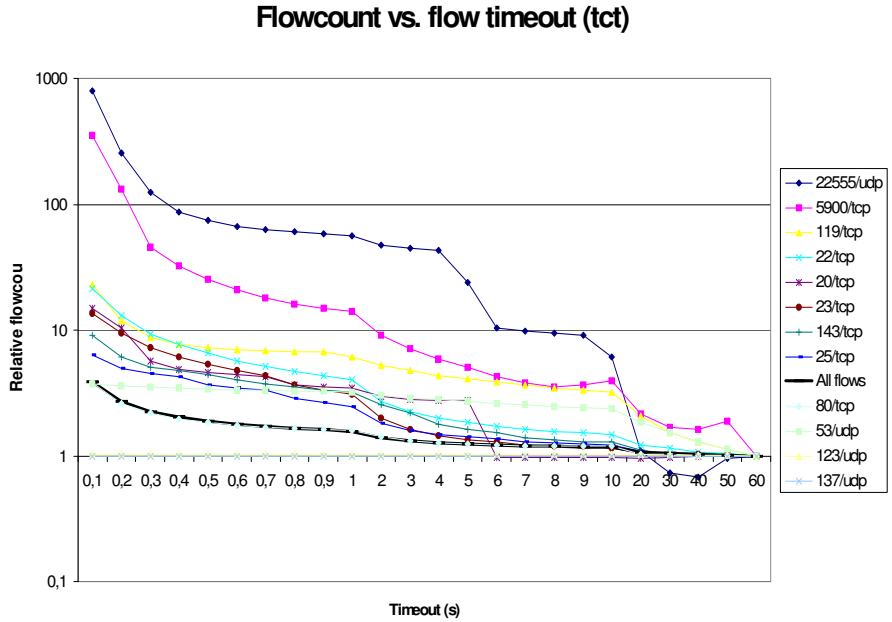
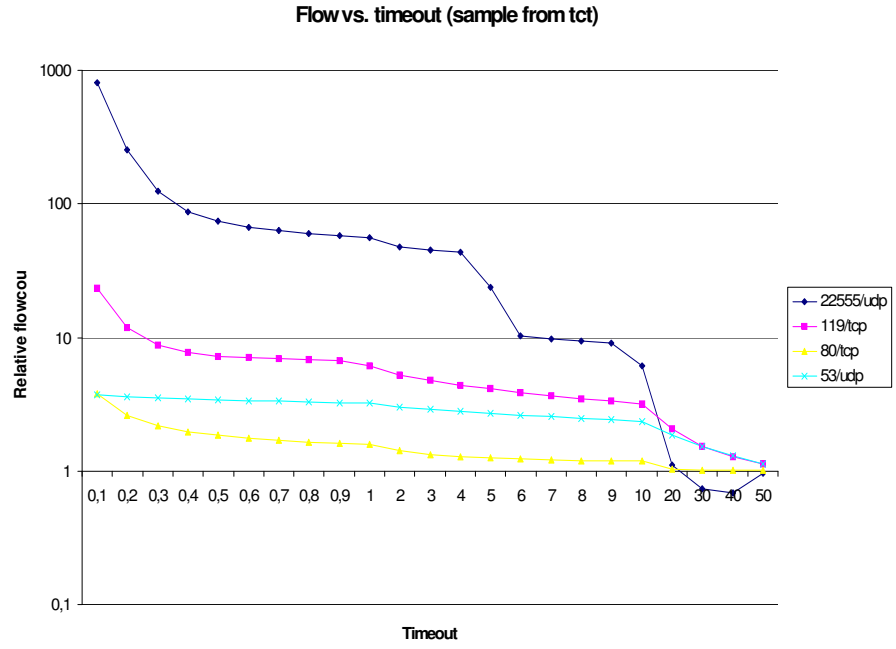
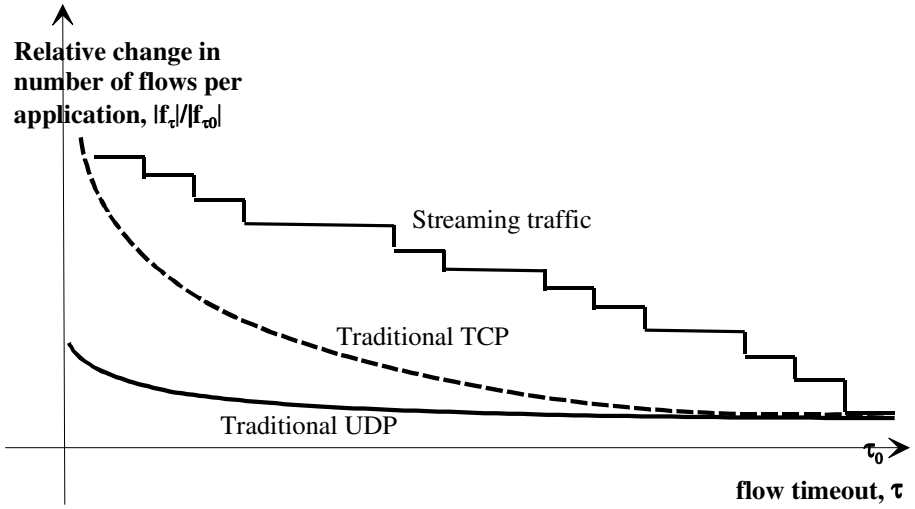


Figure 6.5: Flow count vs. the changing flow timeout in tct-trace

and determine $C_{norm}(a)$ according to eq. 3.9.



(a) Sample from tct-trace



(b) Generalized

Figure 6.6: The effect of flow timeout to the number of observed flows

6.2 Constructing the LVQ classifier for multi-class classification

The process of constructing a classifier to classify the traffic into multiple classes starts by first analyzing, for each application a, the number of flows and the packets on these applications with different flow timeout values. The number of flows (per application) contains information on the usage of this particular application in the network, but since the number of flows is depending on the flow parameters, especially the flow timeout value, it also contains information on the temporal behavior of the application. Therefore,

the flow timeout value is varied in order to differentiate traffic. The application identifier, the number of flows and the number of packets also form the vectors to which we apply the LVQ algorithm. With the aid of the teaching vectors, we use the LVQ algorithm to form the 2-class traffic classifier that is then applied to the results of flow analysis with the 0.1s, 1s and 10s flow timeouts.

The LVQ classifier is applied in the network to perform multi-class classification as illustrated in Figure 6.7 where the points are indicated where the classifier construction process might be influenced:

1. The parameters of the flow; the level of flow granularity and the flow timeout value.
2. The LVQ parameters; number of codebook vectors, number of learning steps and the learning rate.
3. Construction of teaching vectors from the measurement results.

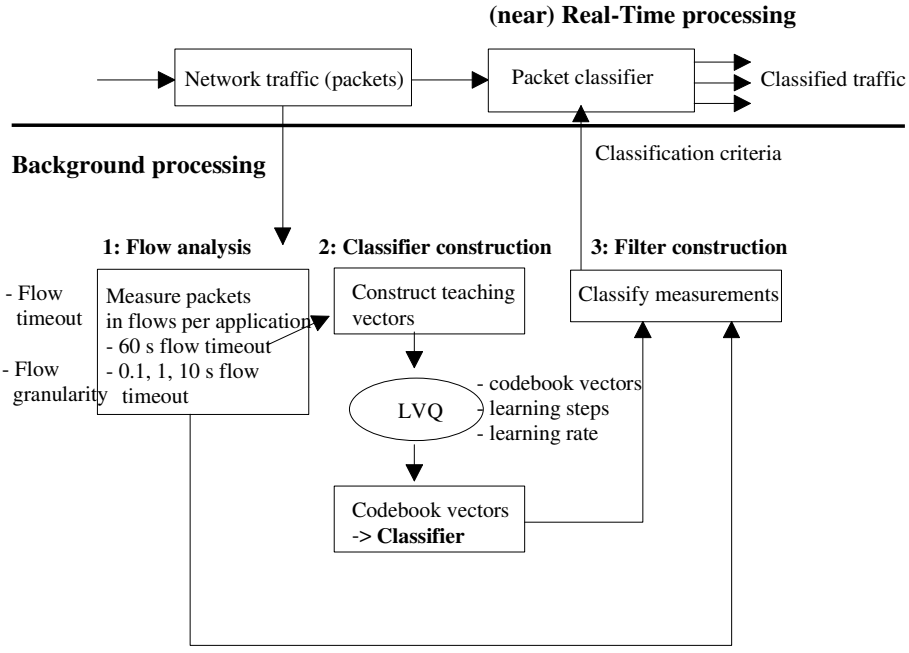


Figure 6.7: Method for applying the LVQ classifier for multiple class classification

We now have a list of all applications together with the total number of packets and flows for each timeout value measured for these applications. Finally, we present these application lists with the measurement results to the classifier that then returns the application profile for each service class. The traffic classification process advances by combining the classification decisions on each timeout value to form a classification string for every application. The applications that are classified to high priority with every flow timeout value are regarded as high-priority applications. All of the applications that receive only default classification decisions are classified to the default class. All other traffic that receives one or two priority decisions, is classified to the priority class.

In this work, for classifying traffic into multiple classes, the goal is set to create policy rules that divide traffic into three classes as shown in Table 6.1 (refer also to the traffic

Table 6.1: Service classes used in the 3-class LVQ traffic classifier

Application class properties in the multi-class LVQ classifier		
Class nr	Intended Class properties	Classification string, [a]
2	High priority traffic, applications that have high presence with smooth flow properties and send packets at very short intervals for lengthy periods of time. The flow count increases as the timeout decreases.	$p \wedge p \wedge p$
1	Priority traffic, applications that have high and smooth presence and send part of the time packets at very short intervals, but have moderate intervals between bursts.	$\neg((p \wedge p \wedge p) \vee (d \wedge d \wedge d))$
0	Default traffic, applications that do not produce a significant amount of traffic and send packets at short bursts with longer intervals in between.	$d \wedge d \wedge d$

[a] p stands for prioritized and d for default traffic

division in the 2-class case in Table 5.3). The policy rules aim to divide the traffic so that the highest priority is assigned to applications that have high presence and that tend to send long bursts or streams of traffic. The aim is to divide the traffic into separate classes by observing even more details on the traffic behavior. As mentioned before, any QoS environment must recognize and adjust itself to three basic categories [74]: the externally controlled unidirectional traffic flows, the long held adaptive reliable traffic flows, and the short duration reliable transactions. These categories are aimed to be placed into classes 2, 1, and 0 respectively as shown in Table 6.1.

The pseudo code for this traffic classification process is presented in Table 6.2. The flow classification and flow analysis processes are identical to those in Tables 3.3 and 3.2 and, therefore, only the traffic classification steps are shown in Table 6.2.

6.3 Performance results

6.3.1 Context analysis

In Table 6.3 the performance statistics for the two priority classes in the multi-class LVQ classifier in different networks are shown. We can see that the packet classification factor in the Class 2 (the highest priority class) is significant starting from 26% and getting as high as over 70% of the packets. The amount of applications detected is low, the flow birthrate factor is tolerable and the use of flow space is reasonable, except in ebb115/class 1 where the use of flow space is quite high. We can also see that the relative number of applications classified to different priorities in the traces changes only a little between the networks. It can also be seen that the average number of packets in the first class

Table 6.2: Pseudo code of traffic classification for multi-class LVQ classifier

process Traffic_Classification
Input $\mathbb{X}, \mathbb{P}_{\mathbb{T}}$;
Init $\mathbb{V}$;
  knn-classify $\mathbb{V}$ *with* $\mathbb{X}$;
Optimize placement of $\mathbb{V}$;
  $v_i[j+1] = [1 - s[j]\alpha_i[j]]v_i[j] + s[j]\alpha_i[j]x[j]$;

Flow_Analysis($\mathbb{P}_{\mathbb{T}}$);
  $\forall\tau$ *Return* $\{(a, F_{norm}(a), C_{norm}(a))\}$;

Classify measurements;
  $\forall\tau$ *Return* $\mathbb{S}_c^{tmp}(\tau) = \{(a), (\arg\min_c \|(C_{norm}(a), F_{norm}(a)) - v_c\|)\}$;
  Return $\mathbb{S}_2 = \mathbb{S}_c^{tmp}(\tau_1, c=1) \wedge \mathbb{S}_c^{tmp}(\tau_2, c=1) \wedge \mathbb{S}_c^{tmp}(\tau_3, c=1)$;
  Return $\mathbb{S}_0 = \mathbb{S}_c^{tmp}(\tau_1, c=0) \wedge \mathbb{S}_c^{tmp}(\tau_2, c=0) \wedge \mathbb{S}_c^{tmp}(\tau_3, c=0)$;
  Return $\mathbb{S}_1 = \neg(\mathbb{S}_1 \wedge \mathbb{S}_3)$;
end Traffic_Classification;

is higher than in the second class and it is not unusual to have over 50% of the packets assigned to the first class.

In Figure 6.8 we observe the per-class context performance statistics of the multi-class LVQ for the dec-networks. We see that in the first class (Figure 6.8) the classifier picks out the applications containing a substantial amount of packets. The applications seem to contain long flows on average since the use of the connection setup resources is quite low. When observing the performance of the second class classification process we see a significant change to the first class behavior. The use of flow space and the flow birthrate are varying and the packet classification level is significantly lower than in the first class.

In Figure 6.9 we observe the per-class performance statistics of the multi-class LVQ classifier in the quadrilateral model for the ebb-networks. For the first class (Figure 6.9) the packet classification factor is high and we can also observe a relatively controlled use of flow space. For the second class the drop in the packet classification factor is clear compared to that of the first class. To summarize, the multi-class classifier seems to produce somewhat ambiguous context performance: The application factor is consistently quite low in both of the priority classes in all of the traces. Otherwise no clear properties can be resolved for class context performance.

6.3.2 Content analysis

The content analysis statistics are shown in Tables 6.4 and 6.5. We can see, that applications in Classes 2 and 1 (classes with highest priority) behave the same way as far as coefficient of variation is concerned. Class 0 (lowest priority) differs significantly. Therefore, the multi-class classification is successful in that it is able to differentiate applications to separate classes. Part of the difference in the coefficient of variation values between Classes 2/1 and Class 0 may be explained by the small number of applications in Class 2 and 1. However, the average behavior of the applications in a class is also a factor to be noted.

In Figures 6.10 to 6.13 we show, how the applications map onto different classes in the

Table 6.3: Basic context performance analysis for the multi-class LVQ classifier

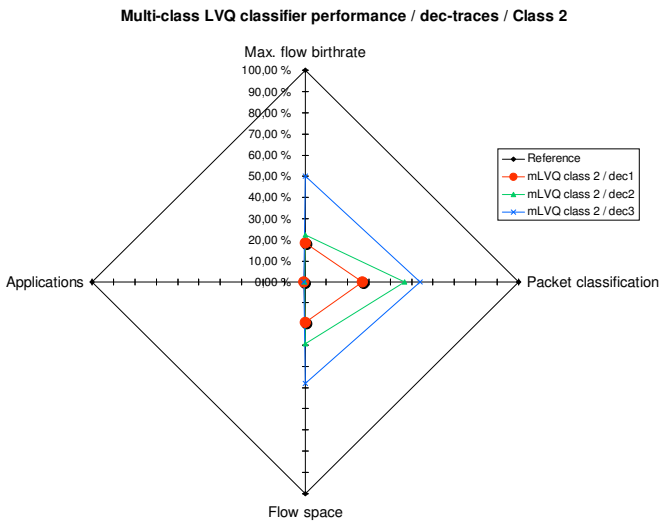
Performance statistics for the multi-class LVQ classifier				
Classified network	Flow birthrate factor, B_{mLVQ}	Flow space factor, S_{mLVQ}	Classification factor, C_{mLVQ}	Application factor, A_{mLVQ}
dec1 / class 2	18,2% at 3566s	19,0%	26,7%	0,64%
dec1 / class 1	20,1% at 3586s	18,9%	21,0%	0,57%
dec2 / class 2	22,1% at 3595s	29,1%	46,5%	0,45%
dec2 / class 1	8,5% at 3583s	9,8%	8,0%	0,78%
dec3 / class 2	50,0% at 3565s	48,0%	54,0%	0,55%
dec3 / class 1	1,0% at 3522s	1,6%	6,6%	0,27%
ebb900 / class 2	22,5% at 3531s	30,5%	64,59%	0,57%
ebb900 / class 1	46,2% at 3610s	41,9%	11,83%	0,49%
ebb115 / class 2	36,5% at 3617s	68,1%	73,82%	1,4%
ebb115 / class 1	1,2% at 3585s	0,6%	3,97%	0,75%
ebb130 / class 2	36,8% at 3573s	24,8%	72,04%	0,91%
ebb130 / class 1	6,0% at 3610s	4,7%	9,08%	1,41%

packet-flow –space.

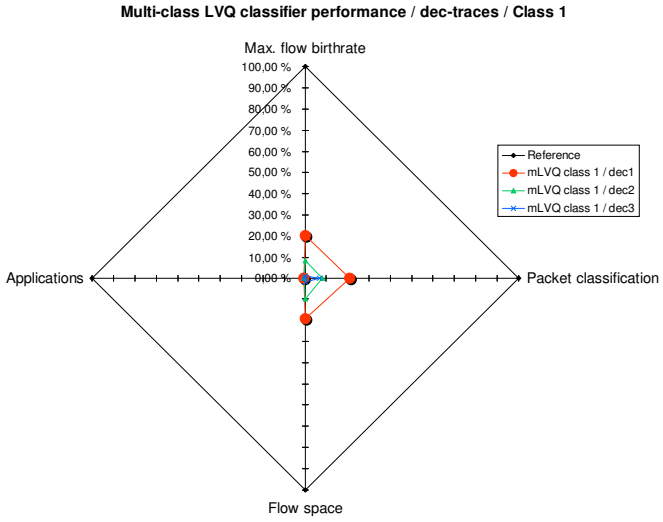
The data points show the application placement with timeout value of 60s only. We also show the application placement with timeout values of 10s, 1s and 0.1s in the packet-flow –space for some hand-picked applications in each class. In general, we can see that the applications having a relatively high packet count against a relatively low flow count are classified higher than applications with opposite characteristics. However, if the flowcount increases significantly (relative to teaching vector placement) the applications are classified to a lower class. This is according to our desired policy rule outcome that aims to limit the classification status of applications with a too high flow count.

When looking at the average application location within the class we notice a significant difference between Class 0 and Class 1 and Class 2. However, a clear difference can not be seen between classes 1 and 2.

In Tables 6.6 and 6.7 we observe the complete per-class network application profiles for each network. Observing Tables 6.6 and 6.7 we can see that there are some consistent



(a) Class 2

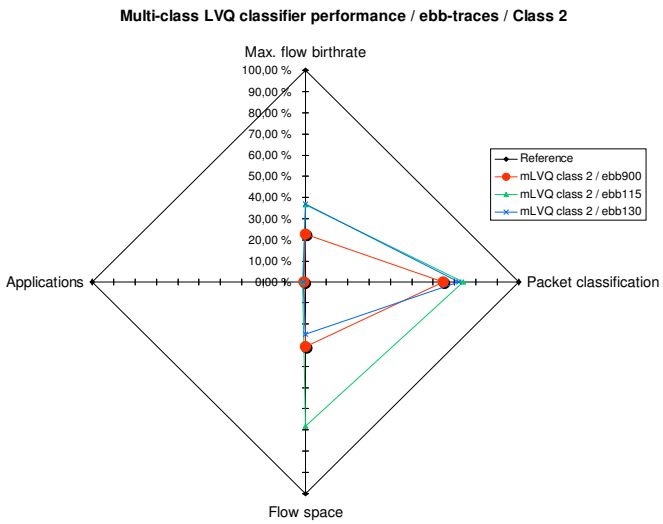


(b) Class 1

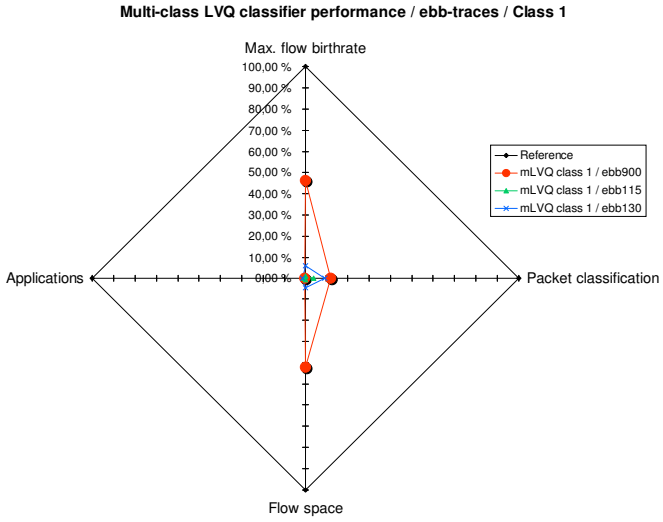
Figure 6.8: Performance of the multi-class LVQ classifier in the dec-traces

candidates for classification but that most of the applications, especially in the higher port-number regions, change from trace to trace. This suggests that in the detailed level the same applications are used varyingly in different networks at different times. The amount of applications classified to the second class is constant, however, the contents of the class change more. Once more, the dec3-trace sees a significant increase in the amount of applications in the first class, while maintaining some of the core applications. This would imply that the application profile is a local phenomenon and its use should be restricted to an autonomous area of the network at the largest.

An interesting phenomenon regarding the network control traffic also occurs in Table 6.7. In all ebb -traces the dns -service (port 53/tcp) is classified to the first class due to a possible dns-database update process. However, at the same time, port 53/udp is classified either to the second or to the third class. All in all, since dns, and similar



(a) Class 2



(b) Class 1

Figure 6.9: Performance of the multi-class LVQ classifier in the ebb-traces

traffic for network control, are classified ambiguously, it is justified to assume that a dedicated class or preferential treatment should be offered for network critical traffic to ensure on-time control functions in the network.

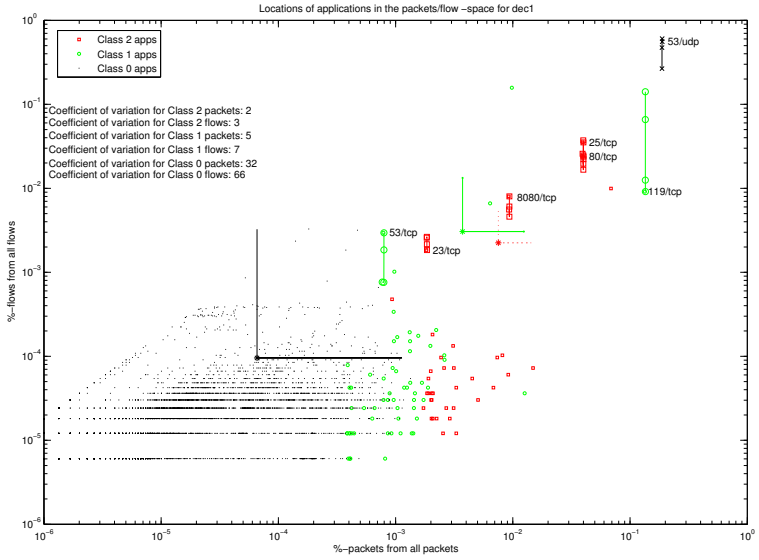
In terminal access protocols the shift from telnet -protocol (port 23/tcp) towards the ssh -protocol (port 22/tcp) is visible as the telnet -service is classified to the first class in all of the dec -traces and in one of the ebb traces and to the second class in two of the ebb -traces while the ssh is getting prioritization to the first and second class in ebb -traces. In addition, the diminishing amount of terminal traffic in general reflects the movement towards the usage of the http -protocol (port 80/tcp). The ftp-data service (port 20) acts consistently throughout all of the networks getting first class classification in all of the traces where present. At least here, the multi-class LVQ classification also proves to be able to distinguish applications that behave in a similar manner over time.

Table 6.4: Content statistics for dec-traces with multiclass LVQ classification

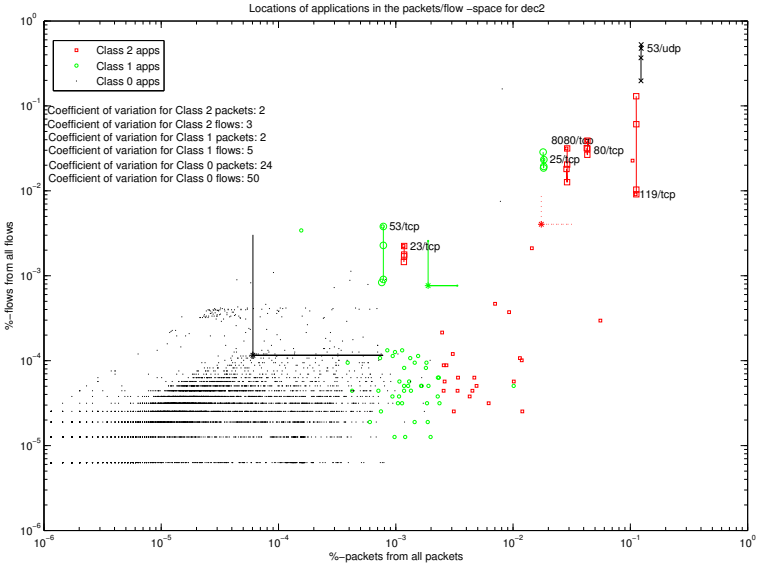
Content statistics for dec-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
dec1	Class 2 pkt	0,7531%	0,1048%	0,09331%	6,8770%	2
dec1	Class 2 flw	0,2244%	0,05%	0,00121%	3,5157%	3
dec1	Class 1 pkt	0,3728%	0,153%	0,03852%	13,3957%	5
dec1	Class 1 flw	0,3053%	0,178%	0,0006%	15,7347%	7
dec1	Class 0 pkt	0,0066%	0,211%	0,00003%	18,6375%	32
dec1	Class 0 flw	0,0096%	0,628%	0,0006%	55,5973%	66
dec2	Class 2 pkt	1,741%	0,156%	0,1165%	11,1124%	2
dec2	Class 2 flw	0,404%	0,058%	0,0025%	3,898%	3
dec2	Class 1 pkt	0,1886%	0,025%	0,01563%	1,808%	2
dec2	Class 1 flw	0,0764%	0,0269%	0,0013%	2,33%	5
dec2	Class 0 pkt	0,0061%	0,139%	0,00003%	12,2226%	24
dec2	Class 0 flw	0,01157%	0,567%	0,0006%	47,5911%	50
dec3	Class 2 pkt	0,001679%	0,0011%	0,000024%	0,0559%	3
dec3	Class 2 flw	0,001091%	0,00040%	0,0003%	0,01773%	1
dec3	Class 1 pkt	0,000974%	0,000232%	0,000024%	0,01525%	2
dec3	Class 1 flw	0,001128%	0,000169%	0,0003%	0,00871%	1
dec3	Class 0 pkt	0,00866%	0,201%	0,00002%	16,0186%	24
dec3	Class 0 flw	0,00869%	0,399%	0,0003%	42,9367%	48

Table 6.5: Content statistics for ebb-traces with multiclass LVQ classification

Content statistics for ebb-traces						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
ebb900	Class 2 pkt	3,114512%	0,1972%	0,607466%	18,155211%	1
ebb900	Class 2 flw	0,612331 %	0,057%	0,023147%	4,841850%	2
ebb900	Class 1 pkt	3,238309 %	0,0669%	1,459591%	5,646093%	1
ebb900	Class 1 flw	5,650641 %	0,1375%	0,526939%	11,182822%	1
ebb900	Class 0 pkt	0,002356 %	0,015%	0,000091%	0,487501%	7
ebb900	Class 0 flw	0,007026 %	0,0541%	0,001362%	4,987541%	9
ebb115	Class 2 pkt	3,270551 %	0,1946%	0,298743%	13,545561%	1
ebb115	Class 2 flw	1,021942 %	0,1061%	0,004444%	10,019256%	2
ebb115	Class 1 pkt	0,448003 %	0,013%	0,231110%	0,713140%	0
ebb115	Class 1 flw	0,102701 %	0,003%	0,020738%	0,268108%	1
ebb115	Class 0 pkt	0,002417 %	0,014%	0,000100%	0,812688%	7
ebb115	Class 0 flw	0,008806 %	0,118%	0,001481%	11,509406%	16
ebb130	Class 2 pkt	3,182031 %	0,215%	0,255221%	16,434920%	1
ebb130	Class 2 flw	0,777384 %	0,111%	0,009284%	11,64816%	3
ebb130	Class 1 pkt	0,841712 %	0,0325%	0,221740%	2,035578%	1
ebb130	Class 1 flw	0,280725 %	0,0129%	0,007427%	0,941434%	1
ebb130	Class 0 pkt	0,002295 %	0,01094%	0,000081%	0,658541%	6
ebb130	Class 0 flw	0,010334%	0,1365%	0,001857%	12,927545%	17



(a) dec1



(b) dec2

Figure 6.10: Locations of priority applications using the multi-class LVQ-classifier / dec1 and dec2

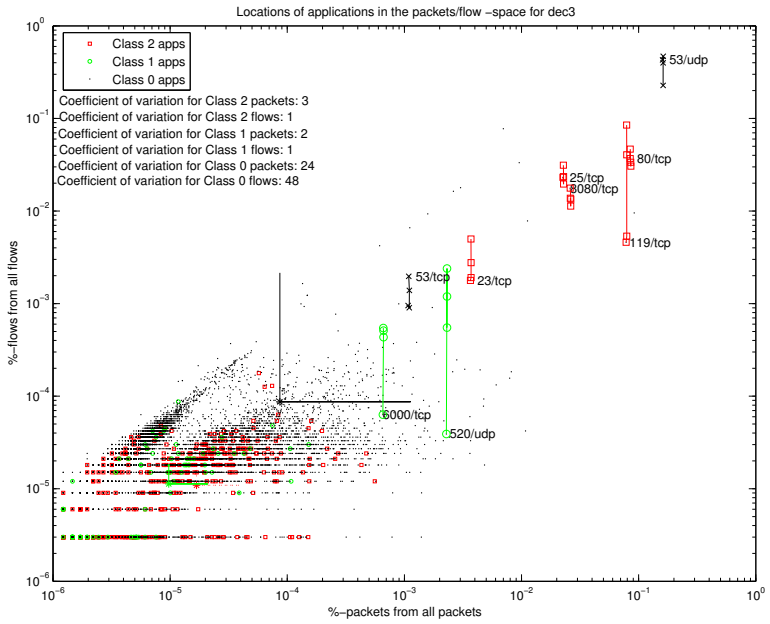
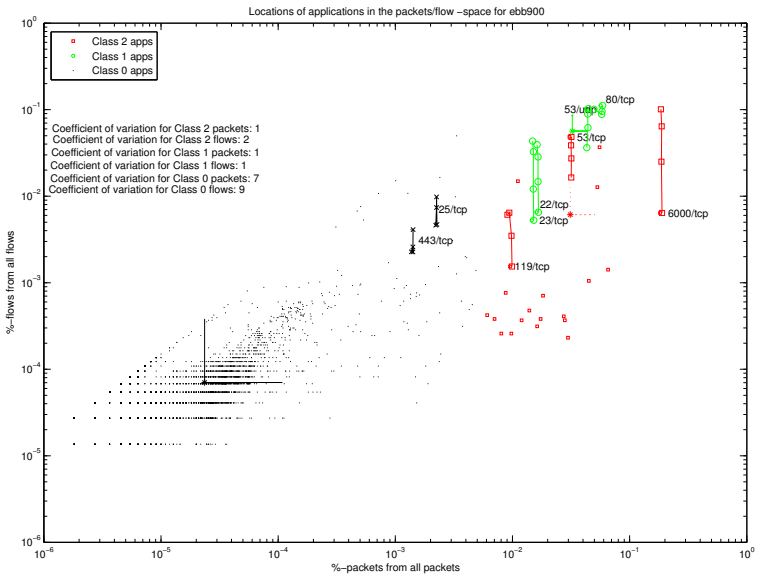
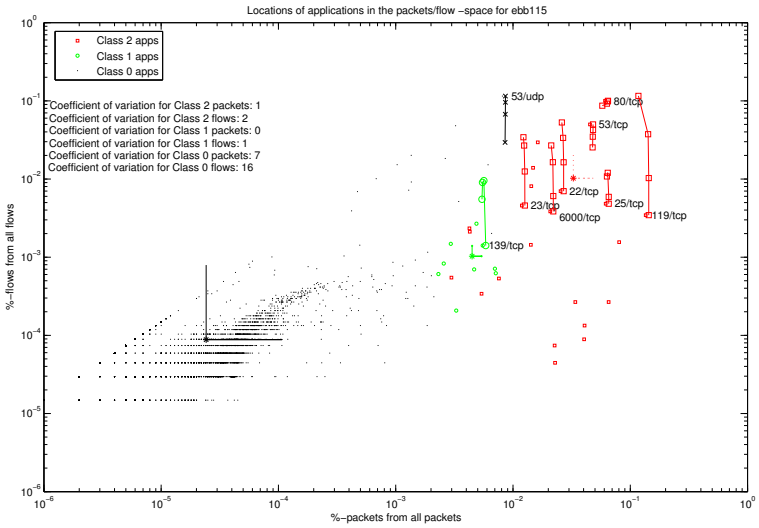


Figure 6.11: Locations of priority applications using the multi-class LVQ-classifier / dec3



(a) ebb900



(b) ebb115

Figure 6.12: Locations of priority applications using the multi-class LVQ-classifier / ebb900 and ebb115

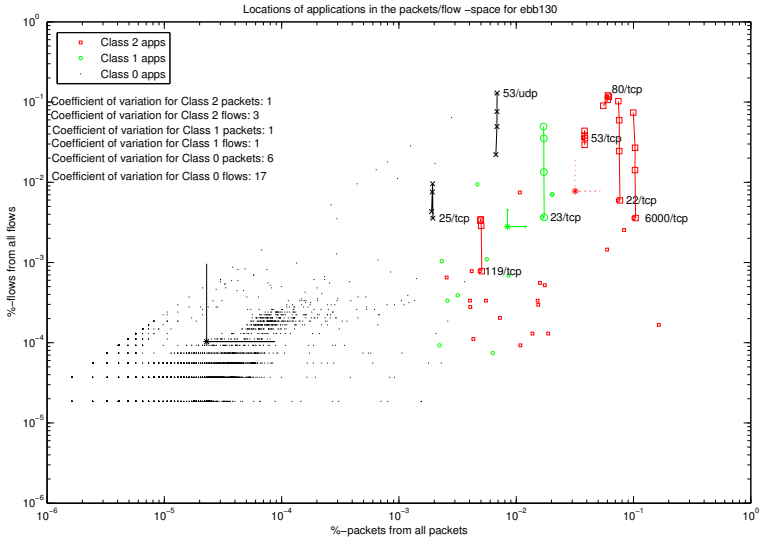


Figure 6.13: Locations of priority applications using the multi-class LVQ-classifier / ebb130

Table 6.6: Priority network application profiles obtained with LVQ classifiers / dec-traces

Complete application lists for multi-class LVQ classifier / dec-traces		
Classified network/class	Application data	Network application profile
dec1 / *class 2*	8 out of 36 applications, 33 tcp, 3 udp	**20/tcp**[a], **23/tcp**, 25/tcp, **80/tcp**, 513/tcp, 520/udp, **1307/udp**, **8080/tcp**
dec1 / *class 1*	6 out of 59 applications, 32 tcp, 17 udp.	21/tcp, 53/tcp, 119/tcp, 123/udp, 540/tcp, **1064/udp**, 5000/tcp
dec2 / *class 2*	7 out of 27 applications, 23 tcp, 4 udp.	20/tcp, 23/tcp, 80/tcp, 119/tcp, 514/tcp, 520/udp, 8080/tcp
dec2 / *class 1*	3 out of 43 applications, 37 tcp, 6 udp.	25/tcp, 53/tcp, 513/udp
dec3 / *class 2*	22 out of 601 applications, 552 tcp, 49 udp.	0/udp, 20/tcp, 21/tcp, 23/tcp, 25/udp, 25/tcp, 70/tcp, 80/udp, 80/tcp, 113/udp, 113/tcp, 119/tcp, 123/udp, 137/udp, 513/tcp, 514/tcp, 540/udp, 6969/tcp, 8000/tcp, 8001/tcp, 8080/udp, 8080/tcp
dec3 / class 1	10 out of 117 applications, 95 tcp, 22 udp.	1/udp, 28/udp, 513/udp, 514/udp, 520/udp, 3001/udp, 3067/tcp, 3310/tcp, 3900/tcp, 6000/tcp

[a]Boldfaced application identifiers indicate applications classified the same way in all dec-traces.

Table 6.7: Priority network application profiles obtained with LVQ classifiers / ebb-traces

Complete application lists for multi-class LVQ classifier / ebb-traces		
Classified network	Application data	Network application profile
ebb900 / *class 2*	6 out of 21 applications, 18 tcp, 3 udp.	0/udp, **53/tcp**[a], **79/tcp**, **119/tcp**, 801/udp, **6000/tcp**
ebb900 / *class 1*	4 applications, 3 tcp, 1 udp.	22/tcp, **23/tcp**[b], 53/udp, 80/tcp
ebb115 / *class 2*	16 out of 23 applications, 18 tcp, 5 udp.	0/udp, 20/tcp, 22/tcp, 23/tcp, 25/tcp, 52/udp, 53/tcp, 79/tcp, 80/tcp, 119/tcp, 719/udp, 794/tcp, 796/tcp, 801/udp, 973/udp, 6000/tcp
ebb115 / *class 1*	2 out of 9 applications, 9 tcp.	139/tcp, 515/tcp
ebb130 / *class 2*	7 out of 23 applications, 23 tcp.	20/tcp, 22/tcp, 53/tcp, 79/tcp, 80/tcp, 119/tcp, 6000/tcp
ebb130 / *class 1*	4 out of 11 applications, 9 tcp, 2 udp.	0/udp, 23/tcp, 110/tcp, 801/udp

[a]Boldfaced application identifiers indicate applications classified the same way in all ebb-traces.

[b]Classified to class 2 only in ebb130.

6.4 Case: Multi-class classification in a multi-service environment

6.4.1 Introduction

We apply the multi-class classification method to a traffic trace measured from the uplink point for the network in the laboratory of Telecommunications technology at the Helsinki University of Technology. We will classify the traffic into three classes by using a set of teaching vectors defined earlier in this work before and by constructing a new set of teaching samples extracted from the trace itself. Having done this, we observe the content and context performance of the classification scheme. The traffic classes we use are shown in Table 6.1. The basic trace properties are shown in Table 5.1. The decision logic for the classification and the simulation environment are the same as described earlier in this chapter. The reference point performance for the tct-trace with the 60 second flow timeout is shown in Table 6.8.

Table 6.8: Performance results at the reference point

Basic simulation results at the reference point with all flows classified				
Network[a]	Flow birthrate factor, B_{base} at time t_F [b]	Flow space, S_{ref} [c]	Classification factor, C_{ref} [d]	Amount of applications, A_{ref}
tct	1600 at 2995998	1425	100%	52462

[a] Refer to table 5.1
[b] referring to the maximum of flow setups in one second in the trace
[c] Maximum number of simultaneous connections during the trace period.
[d] % of packets classified.

Table 6.9: Content statistics for tct-trace with no classification

Content statistics for tct-trace						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
tct	pkt	0,0019%	0,15265%	0,000007%	22,8394%	80
tct	flw	0,0019%	0,125424%	0,00009%	27,9835%	66

The packet/flow -locations of all applications in the trace are shown in Figure 6.14.

6.4.2 Teaching sets

We use two different teaching sets. First one used is the one from ebb900-trace (refer to Figure 5.10) and the other one, based on the examination of the tct-trace, is shown in Figure 6.15. We note that with the tct-trace, we have aimed to pick out the most extreme examples as the prioritized ones: VocalTec (port 22555/udp) is an IP telephone

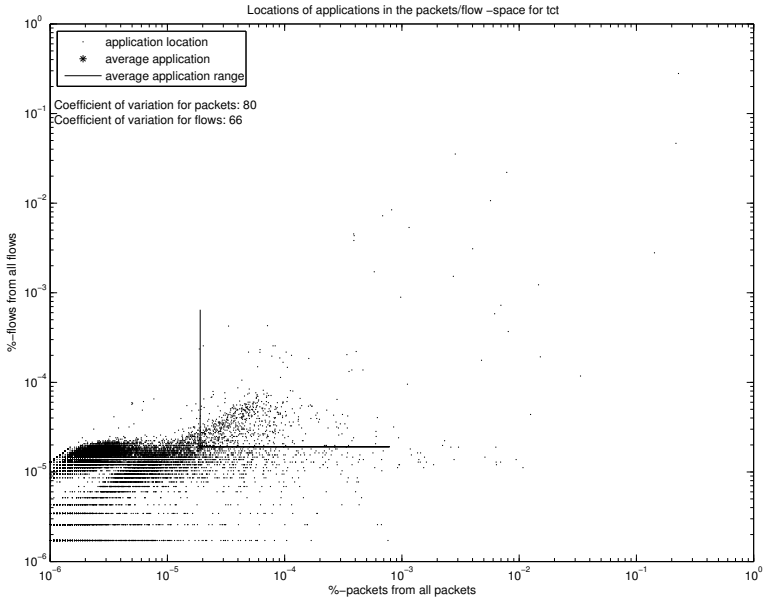


Figure 6.14: Locations of all applications / CASE: tct

application. RealAudio-packets (port 5900/tcp) represent application flows that are of streaming nature. Both of these applications lie in the high packetcount - low flowcount area in the packet-flow –space thus supporting our aim in creating policy rules where flows of high and smooth presence are favored. With the teaching samples for default

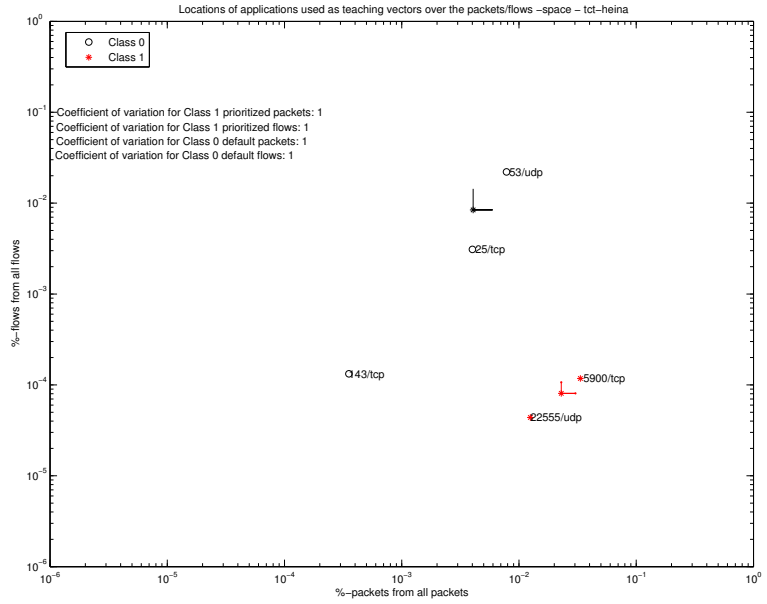


Figure 6.15: Teaching set $\mathbb{X}$ from the tct-trace

treatment we've chosen typical internet applications. The mail-protocols (IMAP and

Table 6.10: Basic context performance analysis for the multi-class LVQ classifier

Performance statistics for the multi-class LVQ classifier				
Classified network	Flow birthrate factor, B_{mLVQ}	Flow space factor, S_{mLVQ}	Classification factor, C_{mLVQ}	Application factor, A_{mLVQ}
tct taught with ebb / class 2	84.3% at 2994522s	99.0%	35.1%	0.03%
tct taught with ebb / class 1	0.13% at 2849666s	0.21%	0.64%	0.01%
tct taught with tct / class 2	2.9% at 2994051s	2.11%	11.25%	0.03%
tct taught with tct / class 1	3.4% at 2998185s	0.90%	11.70%	0.004%

SMTP) are typically used directly by the user and could be considered classical TCP applications. Classifying dns-traffic as default traffic is debatable, since dns-service forms the very foundation over which other services function. However, the aim in picking up the teaching vectors the way we have is to create a classifier that would pick out only a very restricted set of applications with a streaming nature. The classifier construction is identical to the process described earlier in this chapter.

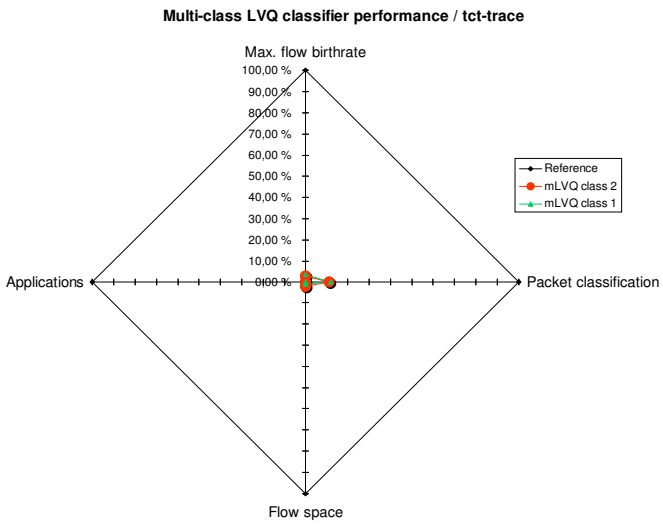
6.4.3 Context analysis

The basic statistics for context analysis are shown in Table 6.10 and in Figure 6.16.

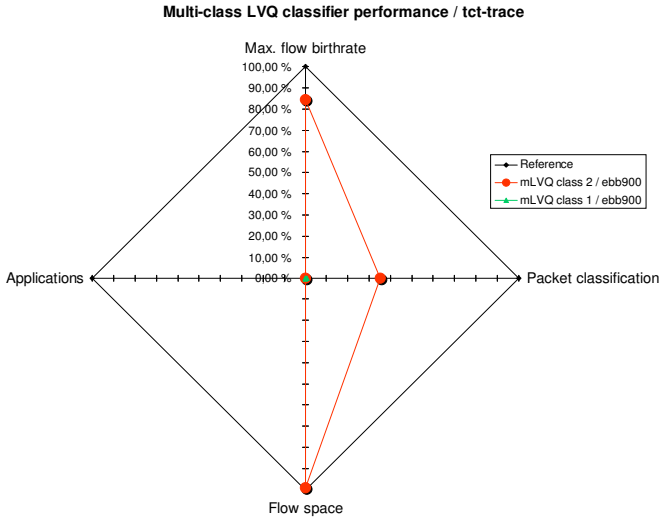
First, we can see from Table 6.10 and Figure 6.16 that the two classifiers perform in a completely different manner. The classifier taught with samples from the tct-network acts very conservatively. It seems that the extreme choices made in choosing the teaching vectors implement the policy rules to an extreme. On the other hand, the classifier taught with samples from the ebb-network uses much more resources, particularly in the first class. In conclusion, both of the classifiers are able to detect a very low proportion of applications that still carry a substantial amount of packets. However, should the multi-class classification be used for router performance optimization, the choice of the teaching vectors and the construction of the classifier should be done with care.

6.4.4 Content analysis

The content analysis statistics are shown in Table 6.11. Once again, based on the observations of the coefficient of variation, the high-priority classes seem to show very similar behavior compared to each other and the low-priority class produces significantly different values of the coefficient of variation. We note that the two priority classes are not distinguishable by the coefficient of variation. Furthermore, since the number of applications in a priority class is very small (2-16 applications) the statistics calculated (mean, standard deviation, minimum, maximum and the coefficient of variation) do not accurately portray the class behavior and characteristics. Once again, great care should



(a) Classifier taught with tct



(b) Classifier taught with ebb900

Figure 6.16: Performance of the multi-class classifier / CASE: tct

be taken to pick out an adequate amount of representative teaching samples to ensure that applications are correctly classified.

From Figure 6.17 we can see that the scarcity of the classified teaching samples leads to an increased overlap between the two classified classes, because the few teaching samples are not able to clearly define the borders of classification in the packet-flow –space. Therefore, the teaching samples should represent the traffic characteristics of each application type and there should be more teaching samples available.

The classified applications (port numbers) in their respective priority classes are shown in Table 6.12 for the tct- and ebb-taught classifiers. The classifier taught with ebb-network classifies many traditional elastic TCP applications to the first class. When comparing the applications chosen to the first and second class with both of the classifiers we see that applications that we have seen before still emerge: the ssh-service (port

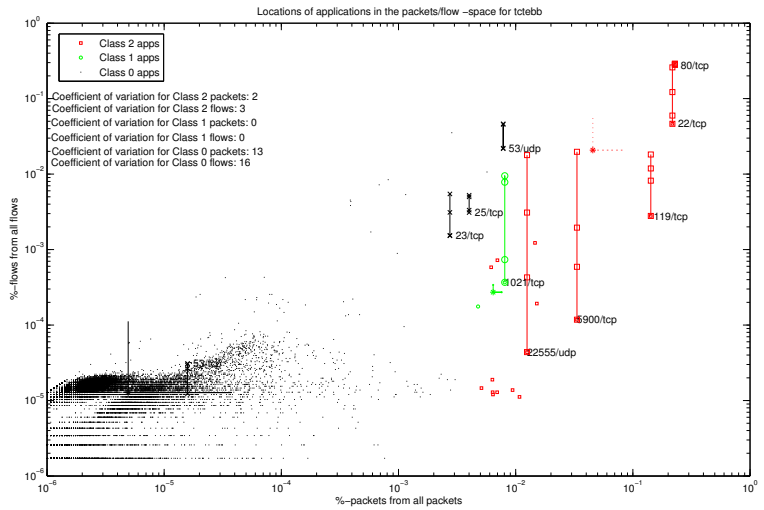
Table 6.11: Content statistics for tct-trace with multiclass LVQ classification

Content statistics for tct-trace						
Trace	Class	mean	stdev	min	max	$\frac{stdev}{mean}$
tct with ebb900	Class 2 pkt	4,552727%	3,731%	22,839412%	0,510189%	2
tct with ebb900	Class 2 flw	2,075225 %	3,088%	27,983511%	0,001117%	3
tct with ebb900	Class 1 pkt	0,642260%	0,1023%	0,806640%	0,477881%	<1
tct with ebb900	Class 1 flw	0,027230%	0,00446%	0,036851%	0,017609%	<1
tct with ebb900	Class 0 pkt	0,039941%	0,1204%	0,783340%	0,000007%	13
tct with ebb900	Class 0 flw	0,122593%	0,468%	3,527875%	0,000086%	16
tct	Class 2 pkt	3,763046%	1,607%	14,246488%	0,941267%	1
tct	Class 2 flw	0,072542%	0,033%	0,279086%	0,001117%	2
tct	Class 1 pkt	7,934532%	2,375%	21,747005%	0,806640%	2
tct	Class 1 flw	1,561330%	0,5065%	4,642759%	0,004381%	2
tct	Class 0 pkt	0,409494%	2,4907%	22,839412%	0,000007%	98
tct	Class 0 flw	0,486283%	3,078%	27,983511%	0,000086%	68

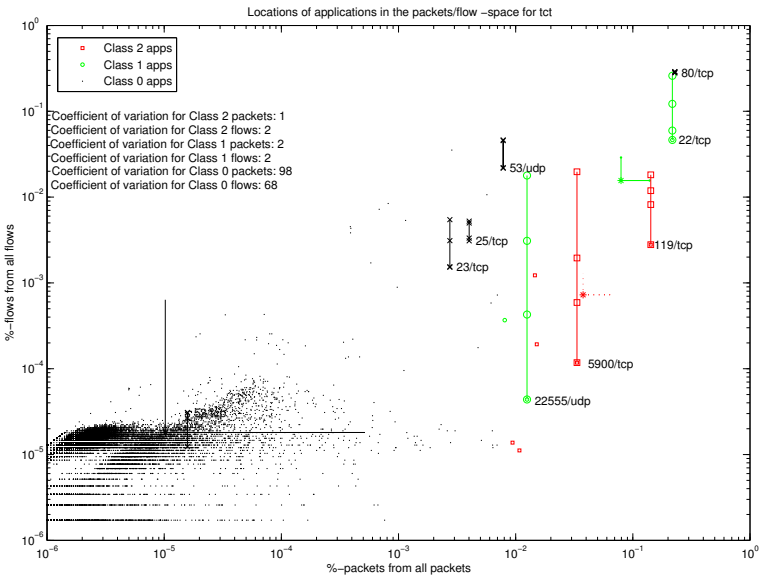
22/tcp) is a candidate for either first or second class classification as well as the ftp-service (port 20/tcp), the news-service (port 119/tcp) and the IP telephone application at port 22555/udp. Looking at the application lists we see that the tct-network teaching samples may not be the best choice, since the amount and type of applications is limited. Especially notable is the classification of the IP-phone application to the second class with the tct-teaching vectors. It would seem, based also on Figure 6.3, that the increase in the flow count with the IP phone application, places the application outside the first class classification. This is further backed up when observing the locations of the priority applications with different timeouts as shown in Figure 6.17. Therefore, the methodology with which the changes in the flow count is applied to classification should be further refined.

6.5 Conclusions

The classification of traffic to multiple classes could be considered reasonably successful from the context performance point of view. There is a large number of packets in a relatively small number of applications in the first class. Particularly, class 1 behavior suggests that a major amount of the packets in the network will be classified to the first class. Consequently, depending on for what classification is used, this might lead to a situation where the whole idea of application differentiation would become meaningless. If a large majority of packets would be prioritized then there would be hardly any differentiation of service levels to different packets, flows, applications and users. However, if



(a) Taught with ebb900



(b) Taught with tct

Figure 6.17: Locations of priority classified applications using the multi-class LVQ-classifier / CASE: tct

the aim is to differentiate similarly behaving traffic, the classification method presented here could be of use. Also remembering, that the teaching vectors of the classifiers were hand-picked with the aim to be able to divide the traffic only into two parts, we may assume that the process of picking up the teaching vectors needs to be enhanced in the multi-class classification case.

We have seen in this chapter that the multi-class LVQ classifier is able to divide the traffic into several classes. Looking from the policy implementation point of view, the contents of these classes present somewhat coherent division of traffic into different classification

Table 6.12: Priority network application profiles obtained with LVQ classifiers / Case: tct / tct

Complete application lists for multi-class LVQ classifier / tct-network		
Classified network	App. data	Network application profile
tct taught with tct / class 2	6 apps.	**119/tcp**[a], **1022/tcp**, **5900/tcp**, **11111/tcp**, **61430/tcp**, **61446/tcp**
tct taught with tct / class 1	3 apps.	22/tcp, **1021/tcp**, 22555/udp
tct taught with ebb / class 2	16 apps.	20/tcp, 22/tcp, 80/tcp, 119/tcp, 1020/tcp, 1022/tcp, 5900/tcp, 11111/tcp, 22555/udp, 61430/tcp, 61446/tcp, 61611/tcp, 61612/tcp, 62934/tcp, 62935/tcp, 62936/tcp
tct taught with ebb / class 1	2 apps.	1021/tcp, 1023/tcp

[a]Boldfaced application identifiers indicate applications classified the same way with both classifiers.

levels with some anomalies. All in all, the network application profiles obtained reflect the nature of the teaching vectors, even though the flow analysis is done using different flow timeout values compared with the value of the flow timeout in the process of selecting the teaching samples.

Cross-referencing Table 6.1 with Tables 6.6, 6.7 and 6.12, we see that the division of traffic into three classes has somewhat ambiguous results in regards to the traffic class contents. Similar applications (like 23/tcp and 22/tcp, the telnet and ssh -services) are classified to different classes in different network environments. Also VoIP-traffic receives different classification decisions depending on the classifier that is used. We can also see that applications like the name services (dns, netbios-ns etc.) sometimes get the second or third class classification. This is unacceptable and network control traffic should be assigned a static class because the name services and other critical traffic are vital for a functional network. These ambiguities may be explained with poor choice and construction of teaching sets and lack of more traffic traces with different traffic types. After all, we are using systems that learn traffic characteristics to classify traffic and if teaching is not done properly results may not be satisfying enough. Although this might be avoided with more careful choice of teaching vectors, it is very probable that the multi-class LVQ classification procedure as a stand alone system is not fit to be used aiming to provide QoS in the network. Nevertheless, the multi-class LVQ classification could be used to provide information to aid in determining the priority traffic while the network critical traffic could be statically classified to an appropriate priority.

We may conclude that the classification is moderately successful and consistent, taking into account the changes and variations in traffic and application usage. Some ambiguity can be detected, however, possibly because of the construction of the original teaching vectors. Special attention should be given towards more clearly detecting hard-interactive applications from the semi-interactive ones and ensuring that network critical traffic is always classified above other traffic. This would require extending and diversifying the classification method.

Once more, we note as we already have noted with the list and the 2-class LVQ classifier that the results for the multi-class classifier contain also the same unrealistic aspect. The application profile was obtained from the same trace on which it was used for analyzing the performance of the classification method. Therefore, in actual deployment the results would be less satisfying than is illustrated in this chapter. However, as we also saw with the 2-class LVQ classifier, the performance degradation is not that big to cloud the interpretation of the results. In a real network we would tackle this by re-calculating the traffic profiles at regular intervals to keep up with the changes in the traffic profile.

Chapter 7

Conclusions

7.1 Results

The goal in this work has been to study novel traffic classification mechanisms based on analysis of traffic measurements to produce traffic policies. The classification methods have been evaluated based on the use of different resources in an IP router and on the application profile these methods produce. A successful automatic detection of applications from the network traffic by the network should result in a policy offering prioritized applications a service level so that the user may be satisfied while simultaneously controlling the use of the network resources. Several methods of application detection, or traffic classification, have been referred to and suggested in this work.

The static application classifier has very good context performance. It carries a lot of packets in a low number of flows and applications. As a result, it uses the IP forwarding resources for prioritized packets very sparingly. This is due to the low amount of classified applications. The prioritized applications are known at all times and provide for stable policy. The downside of this approach is that application lists need to be updated manually to include new application or remove outdated applications from the prioritization status.

The threshold classifier prioritizes applications with high packet/flows -ratio. When the threshold is set low enough (2% in this work) the context performance is optimized. However, if the application lists are not updated frequently enough the use of context performance resources starts increasing. Furthermore, the classified applications may not always be the ones that a user would want to see prioritized and the threshold classifier would not be the optimal choice if user satisfaction would be the goal of the network policy.

When using the 2-class LVQ and multi-class LVQ classifiers, the performance level of the IP router is significantly better compared to the reference point where all flows are classified in a connection oriented setting. The detected application profiles seem to be of reasonable size and the applications detected for priority handling seem to be applications behaving as the policy requires indicating the teaching sample characteristics. However, the network control traffic (name services, for instance) seems to get quite ambiguous treatment. On occasions network critical traffic is classified to priority classes and on

other instances it is not. The same ambiguity is seen with streaming traffic. Based on this, we state that in any network where traffic is differentiated based on traffic measurements there should be separately configured service classes for network critical traffic or traffic with streaming properties. Therefore, the LVQ classifier is mostly suitable for differentiating connectionless applications [75] that require no explicit reservations from the network and connection-oriented applications [75] should be supported with other mechanisms.

In comparison, the threshold classifier could be used as the method to optimize the context performance provided that the update on the application lists is frequent enough. The static application classifier and the LVQ classifier serve as methods to detect applications that directly involve a human at either or both ends of the communication process. The LVQ classifier needs to be carefully taught to detect applications with particular traffic characteristics whereas the static application classifier needs constant updating of the application lists.

The two different types of LVQ classifier presented in this work are able to detect different types of applications from the network and classify them to classes that contain other applications that behave similarly. The classifier prioritizes applications based on their behavior in packet and flow dimensions. The classification is based on measurement results and the resulting application profile is characteristic to the network measured and the teaching data for the classifier. We argue that although the network application profiles change, the behavior of many applications stays the same over time. This phenomenon is seen when a classifier using the LVQ algorithm is taught with a teaching set from one network succeeds in classifying valid interactive applications in another network.

From the performance point of view one classifier produces predictable results in different networks. Therefore the classifiers need not be modified once a classifier with desired properties has been created. However, due to the measurement based nature of the LVQ classifier the classification of the measurement results needs to be done on regular intervals to update the network application profile.

Both of the LVQ classifiers (2-class and multi-class) may be used to aid in creating policies in a network. Since the method to produce policies in this work, is based on characterizing applications using network measurements, we have to look at the possible policies starting from the measured properties[1]. Regardless of the analysis methods that just reduce or filter noise from the measurements, the measurement based policy rules may only be based on the measured properties. Measuring packets and flows and using the measurements together, we can detect four different application types:

1. Application type 1: High on relative flow count- low on relative packet count. This type is an application that has low overall presence and sends bursty, short lived flows with relatively long intervals. Our work has not revealed any clear examples of applications that send this type of traffic.

2. Application type 2: Low on relative flow count - low on relative packet count. This type is an application that has low overall presence with long steady sending of packets. Although densely populated, our work has not revealed any clear examples of applications that send this type of traffic. However, practically any

[1] packets and flows in this work

new emerging application would show up in a traffic trace as this type of an application at the stage when the application use is not widespread.

3. Application type 3: High on relative flow count - high on relative packet count. This type is an application that has high presence and sends bursty, short lived flows with relatively long intervals between flows. Our measurement results show that dns-traffic and, on some occasions, also the http-traffic seems to be of this type.

4. Application type 4: Low on relative flow count - high on relative packet count. This type is an application with high presence long steady sending of packets or frequently recurring transmissions. Our measurements show that typical examples of this type of traffic are the telnet-, ssh-, and news-services. Also applications with streaming traffic properties (RealAudio, VoIP) create this kind of traffic.

Out of these applications we may derive six different scenarios where one type of application is favored over another. More complex scenarios are also possible, for instance, favoring a combination of applications over another combination. However, for simplicity, we will only present cases where two different types of applications are sought out from the traffic traces.

- Scenario a: Application type 1 vs. Application type 3
- Scenario b: Application type 1 vs. Application type 2
- Scenario c: Application type 1 vs. Application type 4
- Scenario d: Application type 2 vs. Application type 3
- Scenario e: Application type 2 vs. Application type 4
- Scenario f: Application type 3 vs. Application type 4

Since application types 1 and 2 may be excluded from the priority application types because of their vague nature, we are left with application types 3 and 4 and, therefore,*Scenario f* is the most interesting one. This scenario could be then mapped, for instance, to Differentiated Services -architecture. Policy rules would be created to place Application type 4 to, for instance, to one of the AF-subclasses whereas other application types would be offered Best Effort -service. Introducing the varying flow timeout data we can further divide Application type 4 into elastic traffic and streaming traffic. This would allow us to put streaming traffic into the EF-class of the Differentiated Services -architecture.

Using the supervised learning methods presented in this work, it is up to the teaching samples to guide the final bias of the policy rule creation based on traffic measurements. We chose in this work to prefer high packet - low flow -types of applications over any other type of applications.

However, the final choice is up to the operator preference following the operator business model. The operators are interested to offer a broader variety of services and to have higher utilization of their network resources [75] to make their business profitable. It is up to the operator to decide what applications receive what kind of treatment in the network. The measurement based system presented in this work only provides information on what

applications behave in which manner and possibly suggestions in which groups or classes the applications should be placed. Therefore, the operator makes the final decisions on application classification based on its view on the business and customers. Propositions for network business models suggest that the operator jump onto the value chain, create added value to commercial providers and reduce their transparency regarding the value created and realized from the Internet. One of the key requirements in doing this is to bind network utilities (e.g. QoS) to commercial providers applications [79] with the support of stringent service level agreements [75]. The scope and focus of this work, however, lie in the analysis of measurements from the network and transport layers and therefore we have not tried to address the economical problems involved above the OSI layer structure. This area should prove to be, however, an ample source for future research.

The use of user and network initiated traffic classification methods are not mutually exclusive. To deploy network initiated traffic classification, however, one must take care that the user needs are always accounted for with the methods of classification and that the classification results make sense. We must also bear in mind that any traffic classification method should not overuse the IP router resources, and consequently cause deteriorating performance also to the user. Therefore, the evaluation of any traffic classification method should include both the user and network perspectives.

Based on the "noise" in the priority application sets, the LVQ classifier should not be used as a network device that is allowed directly to configure the IP packet forwarding equipment. The LVQ classifier should be used more like an expert system that provides network managers filtered information on the types of applications present in the network. The automated creation of classification filters is more like an expert system that gives suggestions and helps in formulating and implementing a policy rather than strictly determines network policy. Furthermore, since malicious users may be able to imitate the packet/flow -behavior of particular applications and thus possibly gain a prioritized status for their traffic, the LVQ classifier, or any other automated measurement based classification method, for that matter, should not be used blindly and without supervision in the network but combined with proper access control and pricing mechanisms. If the final amount of classified applications is very small by a priori knowledge, instead of using dynamic classification methods it might be easier to pick out and update the policy rules manually.

Automated traffic classification using a measurement based approach is still an unsolved issue. In this work, a new solution using the 2-dimensional flow analysis and LVQ algorithm was presented and was found to show some promising results. This implies that the research in this area should advance and also other alternatives in the area of statistical classification should be studied.

7.2 Summary

This work is divided into two parts. In the first part we motivate traffic classification and develop a methodology to assess traffic classification performance both in the connection oriented and connectionless environments. We conclude that there should be more effort to develop classification methods that are also considering the user. The rest of the work is done assessing the performance of previously introduced traffic classifiers and

developing and evaluating a new solution for creating traffic policies, the 2-class LVQ classifier. At the early stages of traffic and flow classifiers the most important thing has been to reduce the amount of resources used in an IP router. Our 2-class and multi-class LVQ classifiers update the priority classified application set according to measurements made from the network, thus being able to adapt to changes in the network traffic and in the use of applications. Our LVQ classifiers are examined in detail and we find them to show moderately controlled and coherent behavior when learning the network traffic characteristics.

7.3 Future research

This thesis has addressed the problem of traffic classification. Traffic classification based on information derived from traffic measurements could be considered to be a part of a network infrastructure that is called a knowledge plane. The basic idea of the knowledge plane in the Internet is to have constructs in the network that provide service or advice to other network elements. The knowledge plain is meant to construct and maintain a high level view of the network and its purpose [80].

Traffic classification is only a small part of the functionality that has to be implemented in the network in order to be able to provide any differentiation of traffic or quality of service. The closely related subject of assigning particular QoS–levels to these differentiated traffic flows is an interesting and demanding issue. To allocate resources appropriately requires knowledge of the (sending) source application behavior. This knowledge may be obtained from the user or it might be determined by analyzing the characteristics of the source. This issue should be pursued further in a separate research effort.

The locality of network traffic also proposes important research issues regarding traffic classification. It should be carefully examined if an application would be prioritized arriving from one part of the network from one ISP's network and advancing to another ISP's network. At the same time the same application arriving or advancing from another part of the network might not earn the priority status. This issue is interesting but hard to carry out because there is a lack of actual traffic traces that have their structure of network topology preserved. This makes it almost impossible to do comparative analysis and other extensive detailed research on topology aware traffic classification. Furthermore, this issue includes studying service level agreements between operators.

Because the method used in this work (measuring the number of packets and flows per application) guides the creation of network application profiles, but the use of network application profiles does not directly guide the total packets/flows -statistics or restrict the use of applications, it might be argued that using a traffic classification scheme like the LVQ classifier directs the usage of applications to the kind that produce large quantities of packets in a few number of flows. To prevent such a phenomenon additional mechanisms of business models and service architectures should be studied. These mechanisms should be applied to prevent users from misusing the network resources.

This work uses very simple statistics obtained from the traffic classes (mean, variance, and the coefficient of variation) to assess the content of the traffic classes. These seem to provide basic and adequate information on the class behavior. However, the efficiency with which the classifiers select applications that actually behave similarly is not clear. Also the classifiers seem to produce slightly different statistics on the default traffic class

(Class 0). Therefore, the evaluation of traffic classifiers, especially the content evaluation, deserves a separate research effort.

The LVQ classifier is a measurement based classifier. It suffers from the learning delay meaning that it has to gather data to adapt to the application profile of the network it is connected to. This initial process requires time, but after that, the network application profile determination can be continuous ensuring that the most recent and precise network application profile is in use. In this work, the learning time is the total length of the trace, approximately 60 minutes for all traces but one. The learning time is, however, an issue that should also be further studied. Subsequently, also the values of learning delay (in this work usually 60 minutes) should be investigated to determine how long does it take for an application to reveal its characteristics in the flow analysis. Our work presented in [81] indicate that in order to obtain a characteristic application profile the length of the measurement period should be more in the order of several hours, possibly even days rather than just a few hours or some tens of minutes. With short measurement periods the changes in the profile statistics are rather abrupt indicating that the formation of policy rules is in a transient state.

The Learning Vector Quantization (LVQ) algorithm is a learning statistical classifying method which suits well to problems involving large amounts of data. The dynamic traffic classifier using the LVQ algorithm provides one classifier alternative. The dynamism in the LVQ classifier is in its capability to react to changing application usage in the network. As in any supervised learning system, the selection of the teaching data requires time and additional effort and makes the classification result subject to the quality of the teaching data: The quality of the LVQ classification is highly dependent on the training data sets. In this work we have selected the teaching samples by hand. However, this should not be the final solution and further research should be directed towards using clustering algorithms as aids for providing teaching samples. The initial teaching set could also be determined using an algorithm to determine from the measurement results which of the applications (or measurement results) belong to particular areas in the packet-flow –space defined by the measurement data. The use of Self Organizing Maps (SOM) is also suggested in [5] and the application of SOM should be an issue of future research when determining the teaching sets. We emphasize that the automatic creation of teaching sets may lead to unintentional built-in policies and therefore teaching sets should always be chosen according to the desired network policy.

The LVQ classifier, as it has been applied in this work, is unable to take into account the limited connection space or flow setup resources in real time, however, in an actual implementation of the LVQ classifier these factors may also need to be accounted for.

Figure 7.1 shows the possible combinations of two measurements with rational aggregation levels (packets that aggregate to flows that aggregate to applications). The small circle markings in Figure 7.1 indicate the work started in [8, 33, 34, 9]. There appears to be other combinations to explore. For instance, it seems that the work done with flows in [41] could be combined with other measurement data to characterize traffic. The possibilities are even further multiplied, if the number of characterizing measurement combinations would be further increased.

The LVQ algorithm itself is designed to include data of several dimensions so one could contemplate on adding several new dimensions to the flow classification application of LVQ. The problem of teaching the classifier that would emerge, is a serious one and may be difficult to overcome. With one, two and maybe even three dimensions the

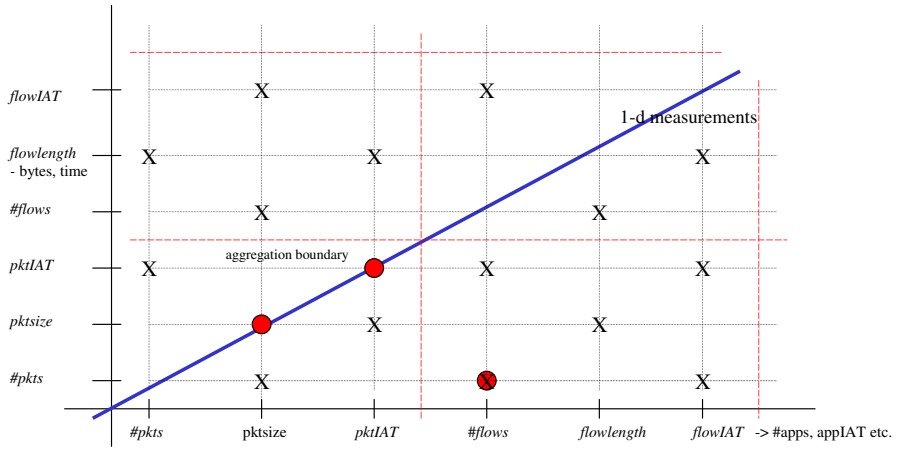


Figure 7.1: Combining measures for traffic characterization

construction of the teaching vectors is feasible. After that it becomes increasingly more difficult as the number of dimensions is increased. The rule of thumb could be that a human handles two dimensions with fluidity, three, if putting some effort into it. Dimensions over three are extremely hard, if not impossible, to handle and comprehend. Therefore, any additional measures, might have to be integrated to the classification system outside the use of the LVQ algorithm.

As Figure 7.1 also shows, there are a lot of other options to measure the network. The measurements would produce multi-dimensional data and the focus would be on finding data clusters that contain applications with similar characteristics. Measurements that are of particular interest are the packet lengths, the packet inter-arrival times within a flow and flow lengths (either in amount of sent data or in time). Also the inter-arrival process of individual flows should be studied to gain knowledge on the flow and application arrival processes.

It should be noted that most of the measurements produce distributions of measured values. The characterization and parameterization of these distributions is a challenging and crucial task: inaccurate or inadequate characterization of a distribution will produce ambiguous or erroneous classification results. Furthermore, as the measurements may be combined, care should be taken to introduce measurements that are as orthogonal to each other as possible. Measurements derived from the original ones with simple calculations or linear operators easily introduce redundancy into the data space. Furthermore, learning from high-dimensional data the right characteristics is more difficult because low data density per dimension requires the user to specify stronger, more accurate constraints on the problem solution, a phenomenon referred to as curse of dimensionality [76].

Bibliography

[1] A. Croll and E. Packman, *Managing Bandwidth - Deploying QoS in Enterprise Networks*, ser. Internet Infrastructure Series. Prentice-Hall, 2000.

[2] V. P. Kumar, T. Lakshman, and D. Stiliadis, "Beyond best effort: Router architectures for the differentiated services of tomorrow's internet," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 152–164, May 1998.

[3] E. Guarene, P. Fasano, and V. Vercellone, "IP and ATM integration perspectives," *IEEE Communications Magazine*, January 1998.

[4] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, September 1995.

[5] T. Kohonen, *Self-Organizing-Maps*, 2nd ed., ser. Springer Series in Information Sciences. Springer–Verlag, 1997.

[6] M. Ilvesmäki, "On the behavior and performance of the packet count flow classifier," *Licentiate thesis*, Helsinki University of Technology, 2001.

[7] T. Worster and A. Doria, "Levels of aggregation in flow switching networks," in *Electronics Industries Forum*. IEEE, 1997.

[8] M. Ilvesmäki and M. Luoma, "On the capabilities of application level traffic measurements to differentiate and classify internet traffic," in *Internet Performance and Control of Network Systems II*, R. D. V. der Mei and F. H.-S. de Bucs, Eds., vol. 4523. SPIE, 2001.

[9] M. Ilvesmäki, M. Luoma, and R. Kantola, "Flow classification schemes in traffic–based multilayer IP switching — comparison between conventional and neural approach," *Computer Communications*, vol. 21, no. 13, pp. 1184–1194, September 1998.

[10] ——, "Learning vector quantization in flow classification of IP switched networks," in *IEEE 1998 Global Telecommunications Conference*, vol. 5, IEEE. IEEE, November 1998, pp. 3017–3022.

[11] M. Ilvesmäki, R. Kantola, and M. Luoma, "Adaptive flow classification in IP switching: The measurement based approach," in *Internet Routing and Quality of Service*, ser. Proceedings of SPIE, R. O. Onvural, Ed., vol. 3529, SPIE. SPIE, November 1998.

[12] M. Ilvesmäki, R. Kantola, and M. Luoma, "Traffic differentiability based on packet and flow per application -analysis," in *Proceedings of Globecom '03*, IEEE. IEEE, 2003.

[13] M. Ilvesmäki and M. Luoma, "Performance analysis of multi-class internet traffic classifier in a connection oriented router environment," in *Internet II: Quality of Service and Future Directions*, ser. Proceedings of SPIE, R. O. Onvural, S. Civanlar, and J. V. Luciani, Eds., vol. Vol. 3842, SPIE. SPIE, September 1999, pp. 70–81.

[14] M. Ilvesmäki, K. Kilkki, and M. Luoma, "Packets or ports - the decisions of IP switching," in *Broadband Networking Technologies*, S. Civanlar and I. Widjaja, Eds., vol. 3233, SPIE. SPIE, November 1997, pp. 53–64.

[15] M. Ilvesmäki and M. Luoma, "IP switching in a simplified ATM environment," in *Broadband Networking Technologies*, S. Civanlar and I. Widjaja, Eds., vol. 3233, SPIE. SPIE, November 1997, pp. 65–76.

[16] J. Apisdorf, K. Claffy, K. Thompson, and R. Wilder, "OC3MON: Flexible, affordable, high-performance statistics collection," MCI Telecommunications Corporation, Tech. Rep., 1997.

[17] "Terminology for policy-based management," RFC 3198, November 2001.

[18] "Policy core information model – version 1 specification," RFC 3060, February 2001.

[19] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area traffic patterns and characteristics (extended version)," *IEEE Network*, November/December 1997.

[20] M. Borden, E. Crawley, B. Davie, and S. Batsell, *Integration of Real-Time Services in an IP-ATM Network Architecture*, August 1995.

[21] S. Shenker, C. Partridge, and G. R., *Specification of Guaranteed Quality of Service*, February 1997.

[22] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, *A Framework for QoS-based Routing in the Internet*, March 1996.

[23] B. Rajagopalan and R. Nair, *Quality of Service (QoS) -based Routing in the Internet - Some Issues*, October 1996.

[24] L. Breslan and S. Shenker, "Best–effort versus reservations: A simple comparative analysis," in *Proceedings of SIGCOMM '98*, ACM. ACM, 1998.

[25] M. Luoma and M. Ilvesmäki, "Measurement based traffic classification in differentiated services," in *Internet Performance and Control of Network Systems II*, R. D. V. der Mei and F. H.-S. de Bucs, Eds., vol. 4523. SPIE, 2001.

[26] K. Kilkki and J. Ruutu, "Selection of QoS mechanisms based on operator's business objective," in *Sixteenth Nordic Teletraffic Seminar - NTS 16*, P. Lassila, E. Nyberg, and J. Virtamo, Eds. Helsinki University of Technology, August 2002, pp. 313–324.

[27] P. Ferguson and G. Huston, *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*. John Wiley and Sons, 1998.

[28] P. P. White, "RSVP and integrated services in the internet: A tutorial," *IEEE Communications Magazine*, vol. 35, no. 5, pp. 100–106, May 1997.

[29] P. Newman, T. Lyon, and G. Minshall, "Flow labelled IP: A connectionless approach to ATM," in *IEEE Infocom, San Francisco*. IEEE, March 1996.

[30] S. Lin and N. McKeown, "A simulation study of IP switching," in *ACM SIGCOMM '97*, 1997.

[31] H. Che, S.-Q. Li, and A. Lin, "Adaptive resource management for IP/ATM hybrid switching systems," in *Broadband Networking Technologies*, S. Civanlar and I. Widjaja, Eds., vol. 3233, SPIE. SPIE, November 1997, pp. 328–339.

[32] P. Newman, G. Minshall, and T. L. Lyon, "IP switching – ATM under IP," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 117–129, April 1998.

[33] B. Nandy, N. Seddigh, A. Chapman, and J. H. Salim, "A connectionless approach to providing QoS in IP networks," in *High Performance Networking*, H. van As, Ed., IFIP. Kluwer Academic Publishers, September 1998, pp. 363–379, iFIP TC-6 Eighth International Conference on High Performance Networking (HPN'98) Vienna, Austria, September 21-25,1998.

[34] A. Chapman and H. Kung, "Automatic quality of service in IP networks," in *Proceedings of the Canadian Conference on Broadband Research*, April 1997, pp. 184–189.

[35] K. Kilkki, *Differentiated Services for the Internet*, ser. Macmillan Technology Series. Macmillan Technical Publishing, 1999.

[36] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: Defining tomorrow's internet," in *Proceedings of SIGCOMM 2002*. ACM, 2002.

[37] G. P. Chandranmenon and G. Varghese, "Trading packet headers for packet processing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 141–151, 1996.

[38] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, September 1999.

[39] R. Jain and S. A. Routhier, "Packet trains - measurements and a new model for computer network traffic," *IEEE Journal on Selected Areas in Communications*, vol. SAC-4, no. 6, September 1986.

[40] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, "A parameterizable methodology for internet traffic flow profiling," *IEEE Journal On Selected Areas In Communications*, vol. 13, no. 8, pp. 1481–1494, October 1995.

[41] K. C. Claffy, "Internet traffic characterization," Ph.D. dissertation, University of California, San Diego, 1994.

[42] F. Baumgartner, T. Braun, and P. Habegger, "Differentiated services: A new approach for quality of service in the internet," in *High Performance Networking*, H. R. V. As, Ed. Kluwer Academic Publishers, 1998, iFIP TC-6 Eighth International Conference on High Performance Networking (HPN'98) Vienna, Austria, September 21-25,1998.

[43] W. Weiss, "QoS with differentiated services," *Bell Labs Technical Journal - Packet Networking*, vol. 3, no. 4, pp. 48–62, October-December 1998.

[44] J. A. Copeland, R. Abler, and K. L. Bernhardt, "IP flow identification for IP traffic carried over switched networks," *Computer Networks*, vol. 31, pp. 493–504, 1999.

[45] S. Keshav and R. Sharma, "Issues and trends in router design," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 144–151, May 1998.

[46] T. Lakshman and D. Stiliadis, "High–speed policy–based packet forwarding using efficient multi–dimensional range matching," in *Proceedings of SIGCOMM '98*, ACM. ACM, 1998.

[47] V. Srinivasan, G. Varghese, S.Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *Proceedings of SIGCOMM '98*. ACM, 1998.

[48] P. Gupta and N. McKeown, "Packet classification on multiple fields," in *Proceedings of SIGCOMM 99*. ACM, 1999.

[49] V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in *Proceedings of SIGCOMM 99*. ACM, 1999.

[50] G. Swallow, "MPLS advantages for traffic engineering," *IEEE Communications Magazine*, pp. 54–57, December 1999.

[51] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communications Magazine*, pp. 42–47, December 1999.

[52] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, "Traffic engineering with MPLS in the internet," *IEEE Network*, vol. 14, no. 2, pp. 28–33, March/April 2000.

[53] M. Loukola, "Data link level forwarding for IPv6 packets on ATM links," *Licentiate thesis*, Helsinki University of Technology, 1997.

[54] H. Che, S.-Q. Li, and A. Lin, "Adaptive resource management for flow–based IP/ATM hybrid switching systems," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 544–557, October 1998.

[55] I. Miller and J. E. Freund, *Probability and Statistics for Engineers*, 3rd ed. Prentice-Hall, 1985.

[56] P. Newman, T. Lyon, and G. Minshall, "Flow labelled IP: Connectionless ATM under IP," in *Networld & Interop, Las Vegas*, April 1996.

[57] A. Kajackas, V. Batkauskas, and A. Medeisis, "Individual qos rating for voice services in cellular networks," *Communications Magazine*, vol. 42, no. 6, pp. 88–93, June 2004.

[58] P. Newman, G. Minshall, T. Lyon, and L. Huston, "IP switching and gigabit routers," *IEEE Communications Magazine*, pp. 64–69, January 1997.

[59] P. Dumortier, "Toward a new IP over ATM routing paradigm," *IEEE Communications Magazine*, January 1998.

[60] M. Ilvesmäki and J. Karvo, "Notes on the per-flow packet count flow classifier," in *Proceedings of Local Computer Network (LCN) 2001*, Accepted, Ed. IEEE, 2001.

[61] J. Zheng, V. O. Li, and X. Yuan, "An adaptive classification algorithm for IP switching," in *Proceedings of Global Telecommunications Conference - Globecom 1999*. IEEE, 1999, pp. 1290–1294.

[62] H. Che and S.-Q. Li, "MPOA flow classification design and analysis," in *Proceedings of Infocom '99*, IEEE. IEEE, 1999, pp. 1497–1504.

[63] W. Wang and C.-C. Shen, "An adaptive flow classification scheme for data-driven label switching networks," in *The IEEE International Conference on Communications (ICC 2001)*, vol. 8 of 10, IEEE. IEEE, 2001, pp. 2613–2619.

[64] S. Taha, H. Che, and S.-Q. Li, "MPOA flow classification design and analysis based on neural network technique," *Computer Communications*, vol. 24, pp. 1007–1018, 2001.

[65] M. Luoma, M. Ilvesmäki, and M. Peuhkuri, "Source characteristics for traffic classification in differentiated services type of networks," in *Internet II: Quality of Service and Future Directions*, ser. Proceedings of SPIE, R. O. Onvural, S. Civanlar, and J. V. Luciani, Eds., vol. Vol. 3842, SPIE. SPIE, September 1999, pp. 25–36.

[66] G. Mueller, P. Sanders, and J. Allen, "Traffic profiles and application signatures," *Computer Communications*, vol. 22, pp. 1123–1126, 1999.

[67] M. Ilvesmäki and J. Karvo, "On the behavior of the candidate table of the per-flow packet count flow classifier," in *Technical Proceedings, 8th IFIP Workshop on the Performance Modelling and Evaluation of ATM & IP Networks(ATM & IP 2000)*, D. D. Kouvatsos, Ed., IFIP. Networks UK Publishers, July 2000.

[68] C. Y. Metz, *IP Switching: Protocols and Architectures*. McGraw - Hill, 1998.

[69] A. Schmidt and D. Minoli, *Multiprotocol over ATM*. Manning Publications Co., 1998.

[70] J. Karvo and M. Ilvesmäki, "Nondeterministic classifier performance evaluation for flow based IP switching," in *High Performance Networking*, H. R. V. As, Ed. Kluwer Academic Publishers, 1998, pp. 613–624, iFIP TC-6 Eighth International Conference on High Performance Networking (HPN'98) Vienna, Austria, September 21-25,1998.

[71] K. ichi Nagami, H. Esaki, Y. Katsube, and O. Nakamura, "Flow aggregated, traffic driven label mapping in label-switching networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1170–1177, June 1999.

[72] E. Kinnunen, "FUNET network traffic analysis - a view of the current state," Master's thesis, Helsinki University of Technology, March 1998.

[73] B. Krishnamurthy, J. C. Mogul, and D. M. Kristol, "Key differences between HTTP/1.0 and HTTP/1.1," *Computer Networks*, vol. 31, pp. 1737–1751, 1999.

[74] P. Ferguson and G. Huston, "Quality of service in the internet: Fact or fiction," 1998.

[75] F. M. Chiussi and A. Francini, "Scalable electronic packet switches," *IEEE Journal On Selected Areas In Communications*, vol. 21, no. 4, pp. 486–500, May 2003.

[76] V. Cherkassky and F. Mulier, *Learning from Data - Concepts, Theory and Methods*, ser. The Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications and Control, S. Haykin, Ed. John Wiley & Sons, 1998.

[77] M. Luoma, "Simulation studies of differentiated services networks," *Licentiate thesis*, Helsinki University of Technology, 2000.

[78] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ_PAK: The learning vector quantization program package," Helsinki University of Technology, Tech. Rep. Report A30, 1996.

[79] B. Krogfoss and J. Pirot, "Next generation networks: enablers for new business models," *Alcatel Telcommunications Review*, no. 2nd quarter, pp. 91–96, May 2001.

[80] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of SIGCOMM 2003*, ACM. ACM, 2003.

[81] M. Ilvesmäki and S. Kaikkonen, "The length of measurement period to determine the application profile for traffic classification in the internet," in *The IEEE International Conference on Communications (ICC 2001)*, vol. 6 of 10, IEEE. IEEE, 2001, pp. 1851–1855.