A Method for Evaluation of Quality of Service in Computer Networks

Tomasz Bujlow, Sara Ligaard Nørgaard Hald, Tahir Riaz, Jens Myrup Pedersen
Section for Networking and Security, Department of Electronic Systems
Aalborg University, DK-9220, Aalborg East, Denmark
tbu@es.aau.dk, slh@es.aau.dk, tahir@es.aau.dk, jens@es.aau.dk

***Abstract*—Monitoring of Quality of Service (QoS) in high-speed Internet infrastructures is a challenging task. However, precise assessments must take into account the fact that the requirements for the given quality level are service-dependent. The backbone QoS monitoring and analysis requires processing of large amounts of data and knowledge of which kinds of applications the traffic is generated by. To overcome the drawbacks of existing methods for traffic classification, we proposed and evaluated a centralized solution based on the C5.0 Machine Learning Algorithm (MLA) and decision rules. The first task was to collect and to provide to C5.0 high-quality training data divided into groups, which correspond to different types of applications. It was found that the currently existing means of collecting data (classification by ports, Deep Packet Inspection, statistical classification, public data sources) are not sufficient and they do not comply with the required standards. We developed a new system to collect training data, in which the major role is performed by volunteers. Client applications installed on volunteers' computers collect the detailed data about each flow passing through the network interface, together with the application name taken from the description of system sockets. This paper proposes a new method for measuring the level of Quality of Service in broadband networks. It is based on our Volunteer-Based System to collect the training data, Machine Learning Algorithms to generate the classification rules and the application-specific rules for assessing the QoS level. We combine both passive and active monitoring technologies. The paper evaluates different possibilities of implementation, presents the current implementation of particular parts of the system, their initial runs and the obtained results, highlighting parts relevant from the QoS point of view.**

***Index Terms*—broadband networks, data collecting, Machine Learning Algorithms, performance monitoring, Quality of Service, traffic classification, volunteer-based system.**

Manuscript received August 1, 2012. This work was a part of a PhD project (Grant No. 8–10100) co-financed by Center for TeleInFrastruktur at Aalborg University, a global research center (see http://www.ctif.aau.dk/), Vækstforum Nordjylland, a regional development project (see http://www.rn.dk/RegionalUdvikling/Vaekstforum/), and Bredbånd Nord A/S, a regional fiber networks deliverer (see http://www.bredbaandnord.dk/).

Tomasz Bujlow is with Section for Networking and Security, Department of Electronic Systems, Aalborg University, Aalborg East, 9220 Denmark (tel: +45 5026 2494, e-mail: tbu@es.aau.dk).
Sara Ligaard Nørgaard Hald is with Section for Networking and Security, Department of Electronic Systems, Aalborg University, Aalborg East, 9220 Denmark (e-mail: slh@es.aau.dk).
Tahir Riaz is with Section for Networking and Security, Department of Electronic Systems, Aalborg University, Aalborg East, 9220 Denmark (e-mail: tahir@es.aau.dk).
Jens Myrup Pedersen is with Section for Networking and Security, Department of Electronic Systems, Aalborg University, Aalborg East, 9220 Denmark (e-mail: jens@es.aau.dk).

I. Introduction

This journal paper is the extended and revised version of [1] which was presented at the 14th International Conference on Advanced Communication Technology (ICACT 2012).

One of the most interesting challenges in today's world is how to measure the performance of computer network infrastructures, when different types of networks are merged together. In the last few years the data-oriented networks evolved into converged structures, in which real-time traffic, like voice calls or video conferences, is more and more important. The structure is composed of traditional data cable or more modern fiber links, existing Plain Old Telephone Service (POTS) lines used to provide analog services (voice telephony), or digital services (ADSL, PBX, ISDN), and nowadays also of mobile and wireless networks. There are numerous methods for the measurement of Quality of Service (QoS) in current use, which provide the measurements both on the user side and in the core of the network. Internet Service Providers are interested in centralized measurements and detecting problems with particular customers before the customers start complaining about the problems, and if possible, before the problems are even noticed by the customers.

Each network carries data for numerous different kinds of applications. QoS requirements are dependent on the service. The main service-specific parameters are bandwidth, delay, jitter, and packet loss. Regarding delay, we can distinguish strict real time constraints for voice and video conferences, and interactive services from delivery in relaxed time frame. In conversation, a delay of about 0.1 s is hardly noticeable, but 0.25 s delay means an essential degradation of transmission quality, and more than 0.4 s is considered as severely disturbing [2].

Therefore, in order to provide detailed information about the quality level for the given service in the core of the network, we need to know, what kind of data is flowing in the network at the present time. Processing all the packets flowing in a high-speed network and examining their payload to get the application name is a very hard task, involving large amounts of processing power and storage capacity. Furthermore, numerous privacy and confidentiality issues can arise. A solution for this problem can be use of Machine Learning Algorithms (MLAs), which use previously generated decision rules, which are based on some statistical information about the traffic. In our research, we used one of the newest MLAs - C5.0. MLAs need very precise training sets to learn how to accurately classify the data, so the first issue to be solved was to find a way to collect high-quality training statistics.

In order to collect the necessary statistics and generate training sets for C5.0, a new system was developed, in which the major role is performed by volunteers. Client applications installed on their computers collect the detailed information about each flow passing through the network interface, together with the application name taken from the description of system sockets. Information about each packet belonging to the flow is also collected. Our volunteer-based system guarantees precise and detailed data sets about the network traffic. These data sets can be successfully used to generate statistics used as the input to train MLAs and to generate accurate decision rules.

The knowledge about the kind of application to which the traffic belongs obtained from MLAs can be used together with traffic requirements for the given application to assess the QoS level in the core of the real network. The real traffic needs to be sampled to obtain the necessary raw statistics. Parameters like jitter, burstiness, download and upload speed can be assessed directly on the basis of information obtained from the captured traffic. To assess delay and packet loss, active measurement techniques must be involved (like ping measurements in both directions).

The remainder of this document is split into several sections, which describe in detail the system architecture and some parts of the implementation. Section II contains the overview of current methods of assessing the network QoS level. Both passive and active methods are described along with their advantages and weaknesses. Section III gives an overview of our methods, so the reader is able to understand how the particular components are built and connected with each other. Section IV describes current methods used for traffic classification in computer networks and it explains why they are not sufficient for our needs. Section V presents our new tool used for collecting and classification of the network traffic – the Volunteer-Based System (VBS). Section VI shows how the statistical parameters are obtained from the data collected by VBS. Section VII evaluates different Machine Learning Algorithms and shows why we chose C5.0 to be included in our system. Section VIII demonstrates design and implementation of the system, while Section IX summarizes the most important points.

II. Related work

During the last 20 years we have been witnesses to the subsequent and increasing growth of the global Internet and the network technology in general. The broadband and mobile broadband performance today is mainly measured and monitored by speed. However, there are several other parameters, which are important for critical business and real-time applications, such as voice and video applications or first-person shooter games. These parameters include download and upload speeds, round trip time, jitter, packet loss, and availability [3], [4].

The lack of the centralized administration makes it difficult to impose a common measurement infrastructure or protocol. For example, the deployment of active testing devices throughout the Internet would require a separate arrangement with each service provider [3]. This state of affairs led to some attempts to make simulation systems representing real characteristics of the traffic in the network. Routers and traffic analyzers provide passive single-point measurements. They do not measure performance directly, but traffic characteristics are strongly correlated with performance. Routers and switches usually feature a capability to mirror incoming traffic to a specific port, where a traffic meter can be attached. The main difficulty in passive traffic monitoring is the steadily increasing rate of transmission links (10 or 100 GB/s), which can simply overwhelm routers or traffic analyzers, which try to process packets. It forces introduction of packet sampling techniques and, therefore, it also introduces the possibility of inaccuracies. Even at 1 Gbit/s, the measurement can result in enormous amount of data to process and store within the monitoring period [3].

To overcome the heavy load in the backbone and to not introduce inaccuracies, a smart monitoring algorithm was needed. There are several approaches to estimate which traffic flows need to be sampled. Path anomaly detection algorithm was proposed in [5]. The objective was to identify the paths, whose delay exceeds their threshold, without calculating delays for all paths. Path anomalies are typically rare events, and for the most part, the system operates normally, so there is no need to continuously compute delays for all the paths, wasting processor, memory, and storage resources [5]. Authors propose a sampling-based heuristic to compute a small set of paths to monitor, reducing monitoring overhead by nearly 50 % comparing to monitoring all the existing paths.

The next proposals on how to sample network traffic in an efficient way were made on the basis of adaptive statistical sampling techniques, and they are presented in [6] and [7].

If a congestion is detected, from user's perspective it is very important to know, if the congestion is located on the local or on the remote side. If the link experiences a local congestion, the user may be able to perform certain actions, e.g. shut down an application, which consumes a lot of bandwidth, to ease the congestion. On the other hand, if the congested link is a remote link, either in the Internet core or at the server side, the back-off of the low-priority applications on the user's side is unnecessary. It only benefits the high-priority flows from other users, which compete for that link. Since this altruistic behavior is not desirable, the low priority TCP only needs to back off, when the congested link is local [8].

Detecting the location of congestion is a challenging problem due to several reasons. First of all, we cannot send many probing packets, because it causes too much overhead, and it even expands the congestion. Secondly, without router support, the only related signals to the end applications are packet losses and delays. If packet losses were completely synchronized (packets were dropped from all the flows), the problem would be trivial. In reality, the packet loss pattern is only partially synchronized [8]. Authors of [8] attempted to solve the problem of detecting the location of the congestion by using the synchronization of the behavior of loss and delay across multiple TCP sessions in the area controlled by the same local gateway. If many flows see synchronized congestion, the local link is the congested link. If the congested link is remote, it is less likely that many flows from the same host pass the same congested link at the same time. If there is only a small number of flows which see the congestion, the authors performed an algorithm based on queuing delay patterns. If the local link is congested, most flows typically experience high delays at a similar level. Otherwise, the congestion is remote [8].

Traffic can be profiled according to the protocol composition. Usually, predominance of the TCP traffic is observed (around 95 % of the traffic mix). When congestion occurs, TCP sources respond by reducing their offered load, whereas UDP sources do not. It results in the higher ratio of UDP to TCP traffic. If the proportion becomes high and the bandwidth available to TCP connections becomes too low to maintain a reasonable transmission window, the packet loss increases dramatically (and TCP flows become dominated by retransmission timeouts) [3]. Packet sizes provide insight into the type of packet, e.g. short 40-44 bytes packets are usually TCP acknowledgment or TCP control segments (SYN, FIN or RST) [3].

Active methods for QoS monitoring raise three major concerns. First, the introduction of the test traffic will increase the network load, which can be viewed as an overhead cost for active methods. Second, the test traffic can affect measurements. Third, the traffic entering ISP can be considered as invasive and discarded or assigned to a low-priority class [3].

Within an administrative domain (but not across the entire Internet), performance can be actively monitored using the data-link layer protocol below IP, as the Operations, Administration and Maintenance (OAM) procedure in ATM and MPLS networks. As a result, at the IP layer it is often desirable to measure performance using the IP/ICMP protocol. So far, most tools or methods are based on ping (ICMP echo request and echo reply messages) or traceroute (which exploits the TTL field in the header of the IP packet) [3].

Although the round-trip times measured by ping are important, ping is unable to measure the one-way delay without additional means like GPS to synchronize clocks at the source and destination hosts. Another difficulty is that pings are often discarded or low-prioritized in many ISP networks. Traceroute will not encounter this problem because UDP packets are used. However, traceroute has known limitations. For example, successive UDP packets sent by traceroute are not guaranteed to follow the same path. Also, a returned ICMP message may not follow the same path as the UDP packet that triggered it [3].

Although end-to-end performance measurements can be carried out at the IP layer or the transport/application layer, the latest is capable of measurements closer to user's perspective. The basic idea is to run a program emulating a particular application that will send traffic through the Internet. All the parameters (delay, loss, throughput, etc) are measured on the test traffic. This approach has one major drawback - custom software needs to be installed at the measurement hosts [3].

On the basis of the mentioned work we found out that the existing solutions are not sufficient for precise QoS measurements. This state of affairs motivated us to create a new system which combines both passive and active measurement technologies.

III. The overview of the methods

The flow chart of the system is shown in Figure 1. The following paragraphs contain detailed description of our methods. At first, the volunteers must be recruited from the network users. The volunteers install on their computer a client program, which captures relevant information about the traffic and submits the data to the server. On the server these data are used to generate per-application traffic statistics. The C5.0 Machine Learning Algorithm uses these statistics to learn how to distinguish between different types of applications and, later, it generates the classification rules (decision trees).

In order to assess the network QoS level in the core of the network for particular users we needed to find a method to capture the relevant traffic. The challenging task is to process significant amount of traffic in the high-speed networks. When the relevant flows are captured, per-flow statistics need to be generated. There are two kind of statistics generated at this step: One used for determining the kind of application associated with that flow, and one used for assessing the QoS level in the passive way. The system uses previously generated classification rules together with the first type of statistics to find out which application the flow belongs to. Then, on the basis of the kind of the application, the system determines ranges of values of the relevant QoS parameters. The last step is to check if the current values (obtained from flow statistics or in the active way) match the expected ones. If not, the quality of the given service is considered as degraded.

IV. The current methods for obtaining pre-classified data

There are many existing methods for obtaining pre-classified data, but none of them were feasible to deliver data required by us to obtain accurate statistics, which could be used to train Machine Learning Algorithms (MLAs). The traffic classification requires the packets to be logically grouped into some structures, which could be assigned to the particular application. The most common used structure among the classification methods is the flow defined as a group of packets, which have the same end IP addresses, ports, and use the same transport layer protocol. In this paragraph we describe the methods and evaluate their usefulness in providing data for our system.

A. Capturing raw data from the network interfaces

The first possibility is to install one application at a time on a host, and to capture its traffic by an external tool, such as Wireshark [9]. Unfortunately, this approach is very slow and it is not scalable. At first, it requires us to install

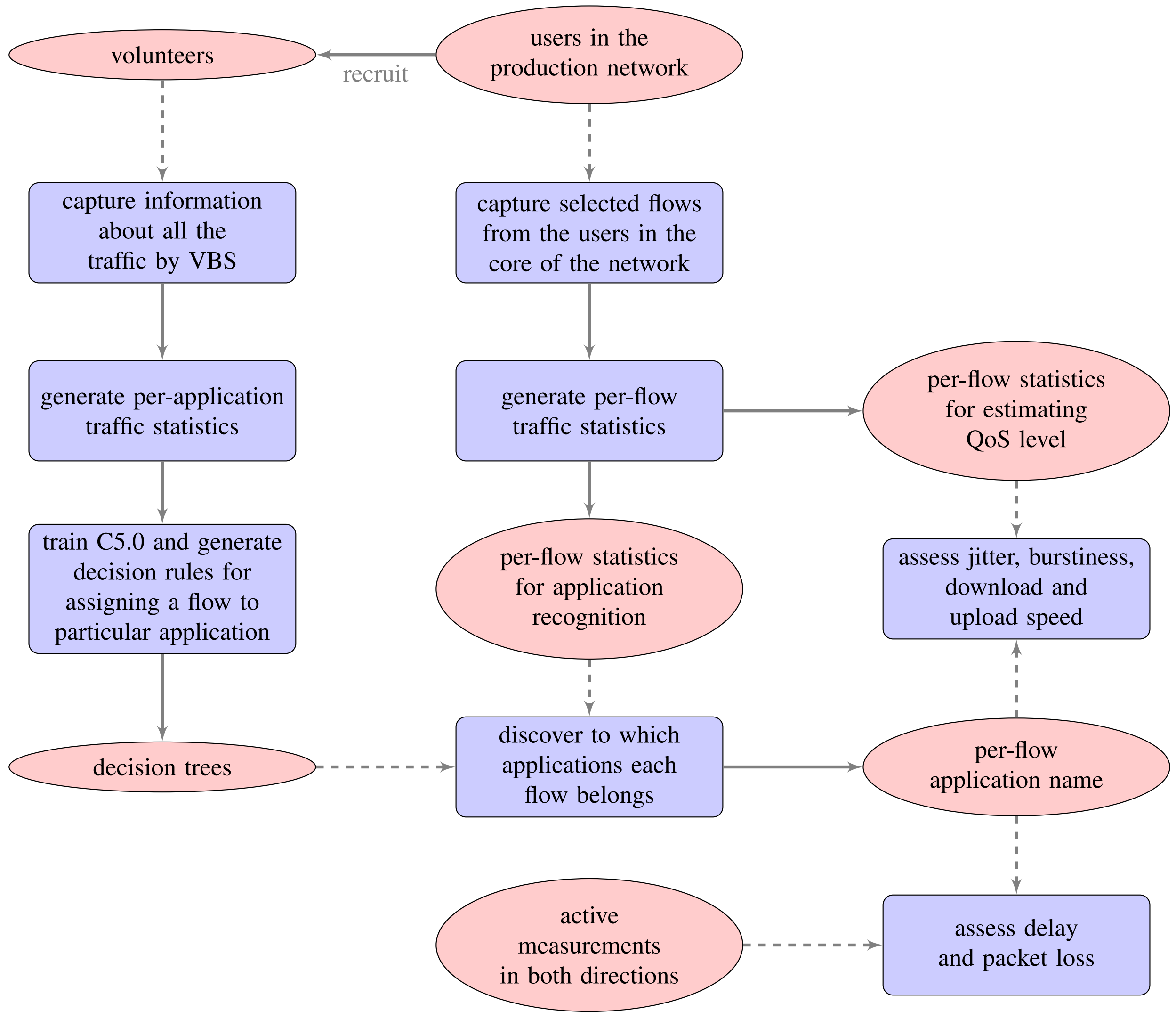
volunteers
recruit
users in the production network
capture information about all the traffic by VBS
capture selected flows from the users in the core of the network
generate per-application traffic statistics
generate per-flow traffic statistics
per-flow statistics for estimating QoS level
train C5.0 and generate decision rules for assigning a flow to particular application
per-flow statistics for application recognition
assess jitter, burstiness, download and upload speed
decision trees
discover to which applications each flow belongs
per-flow application name
active measurements in both directions
assess delay and packet loss

Figure 1. The flow chart of the system

on a host each application that generates traffic we want to capture. Before installing the application, we must uninstall all other applications that can generate any network traffic. The next drawback is that every operating system has some background processes and many of them transmit some data through the network. An example of such a process is the system updater, which can run in background. There is no simple way to recognize packets belonging to the traffic generated by the application intentionally run by us, so the captured sample contains a variable percentage of noise. Finally, some applications, for example, web browsers, can generate various types of traffic. Raw traffic capturers cannot distinguish interactive web traffic, web radio podcasts, video transmissions, or downloads of big files, performed by the same browser.

B. The classification by ports

The port-based classification [10], [11] is very fast, and it is supported on almost all the layer-3 devices in computer networks. Unfortunately, this method is limited to services, protocols, and applications, which use fixed port numbers. It means that with big probability we can correctly classify, for example, traffic generated by e-mail clients and file transfer clients using File Transfer Protocol (FTP), when they use the default ports to connect to servers. However, even in this case we have false positives and false negatives. False negatives result from non-standard ports used in this example by SMTP, POP3, or FTP servers. When a network administrator changes the port used by the given service (due to security reasons), the traffic is not classified correctly. False positives result from malicious applications, which intentionally use some well-known port numbers to be treated in the network with a priority, or to be able to transmit data at all. Such situation exists when a Torrent user runs his client on port 80, which cause the traffic to be treated as if it originated from a web server. Another big concern of port-based classification is the inability of recognizing different types of traffic using the same transport-layer protocol and the same transport-layer port. This drawback is strongly visible in the example of HTTP traffic, which can consist of data generated by interactive web browsing, audio and video streaming, file downloads, and HTTP tunneling for other protocols. Finally, the classification made by ports is unable to deal with protocols using dynamic port numbers, like BitTorrent or Skype [9], [12], [13].

C. The Deep Packet Inspection (DPI)

The big advantage of the Deep Packet Inspection (DPI) [14] is the possibility to inspect the content of the traffic. It includes both inspecting particular packets, and inspecting flows in the network as the whole. For that reason, it makes it possible to distinguish different kinds of content generated by the same application, or using the same application-layer protocol, such as HTTP. However, DPI is slow and requires a lot of processing power [9], [12]. Therefore, due to high load in today's network infrastructures, it is not feasible to run DPI in the core of the network. Speed of Internet connections provided to private users tends to increase much faster than processing power of their machines, so performing DPI on user's machines became impossible in my case. Feasibility to perform DPI on the user side does not depend only on possessing the necessary processing power, but also on the user's impression. High CPU usage tends to slow down the machine and it causes additional side-effects, for example, a howling CPU fan. For that reason, full DPI can be done only in a limited number of cases, namely on fast machines using a slow Internet connection. DPI also brings privacy and confidentiality issues, as it can reveal some highly sensitive personal data, such as information about used credit cards, logins and passwords, websites visited by the user, etc [9]. Moreover, DPI is unable to inspect encrypted traffic. Finally, DPI depends on signatures of various protocols, services, and applications, which need to be kept up to date.

D. The statistical classification

Solutions using statistical classification became quite popular during the last few years [14]. To its characteristics we can include fast processing and low resource usage. Statistical classifiers are usually based on rules, which are automatically generated from samples of data. Therefore, such kinds of classifier often make use of Machine Learning Algorithms (MLAs). Apart from all these advantages, statistical classifier have one big drawback – they need to be trained on the samples of data. So the technique assumes that we have already correctly classified data, which we can provide as the input to train the statistical classifier. For that reason, we cannot use this method to collect and classify the initial portion of data.

V. The Volunteer-Based System

The drawbacks of the existing methods for classification of traffic in computer networks led us to the conclusion that we need to design and build another solution. Therefore, we decided to develop a system based on volunteers, which captures the traffic from their network interfaces, and groups the traffic into flows associated with the application name taken from Windows or Linux sockets. The architecture and the prototype were described and analyzed in [15] and [16], and the first version of our current implementation was presented in [17]. Afterwards, the system was extended to support recognizing different kinds of HTTP traffic, and it was named Volunteer-Based System (VBS). The detailed description and evaluation of the extended version of the VBS system can be found in [18]. We released the system under *The GNU General Public License v3.0*, and we published it as a SourceForge project. The project website [19] contains all the information needed to use the system (binary packages, screenshots, documentation and bug tracking system) as well as to perform further development (source code, roadmap, comprehensive documentation of the source code).

The architecture of the system is shown in Figure 2. This cross-platform solution consists of clients installed on users' computers (Microsoft Windows XP and newer and Linux are supported), and of a server responsible for storing the collected data. The client registers information about each flow passing the Network Interface Card (NIC), with the exception of traffic to and from the local network. The captured information are: The start time of the flow, the anonymized identifiers of the local and the remote IP addresses, the local and the remote ports, the transport layer protocol, the anonymized identifier of the global IP address of the client, the name of the application, and the identifier of the client associated with the flow. The system also collects information about all the packets associated with each flow: The identifier of the flow to which the packet belongs, the direction, the size, the TCP flags, the relative timestamp to the previous packet in the flow, and the information about the HTTP content carried by the packet. It is worth mentioning that one flow can contain many higher-layer streams, for example, one TCP flow can contain multiple HTTP conversations. Each of these conversations can transfer different kinds of content, like web pages, audio and video streams, or downloads of big files. For that reason we extract from HTTP headers information necessary to precisely separate the HTTP streams, and we append the information about the type of the stream to the first packet of the stream.

The collected information is then transmitted to the server, which stores all the data in a MySQL database for further analysis. The system was shown in [18] to be feasible and capable of providing detailed per-application information about the network traffic. An example of stored flows on the server side is shown in Table I. The IP addresses for privacy reasons are translated by a one-way hash function and they are stored as anonymized identifiers. The information about the packets belonging to one complete TCP conversation is presented in Table II. As shown, this is an HTTP communication, during which there were transferred two files of the same type with identifier 22 (*text/html*).

The data collected during our experiments by the Volunteer-Based System were used for training the C5.0 Machine Learning Algorithm to be able to recognize traffic generated by different types of applications and different types of traffic. The first approach, focusing on distinguishing 7 different applications (Skype, FTP, torrent, web browser, web radio, America's Army and SSH) and achieving accuracy of over 99 % was described and evaluated in [20]. The second

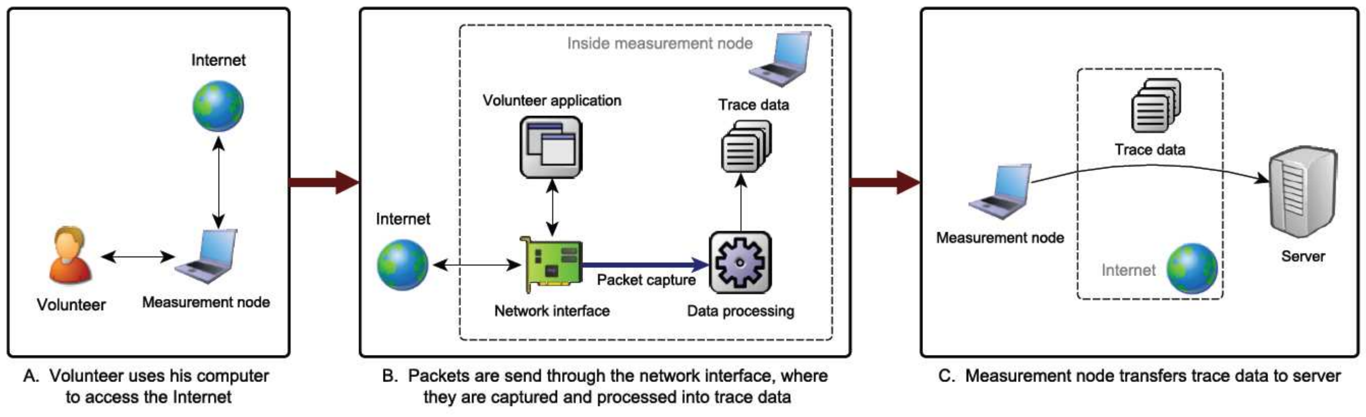
Internet
Volunteer
Measurement node
A. Volunteer uses his computer to access the Internet
Inside measurement node
Volunteer application
Trace data
Internet
Packet capture
Network interface
Data processing
B. Packets are send through the network interface, where they are captured and processed into trace data
Trace data
Measurement node
Internet
Server
C. Measurement node transfers trace data to server

Figure 2. Overview of the VBS system [16]

Table I
EXAMPLE OF THE STORED FLOWS DATA

flow id	client id	start time	local IP	remote IP	local port	remote port	protocol name	global client IP	application name
1193430	4	1325445237826039	d1e0229	fb70266	48293	25395	UDP	178a02f1	uTorrent
2393417	5	1325445237826176	f4c025e	12230296	2276	80	TCP	177d02ef	chrome
1193423	1	1325445237826304	d20022b	11920285	53778	80	TCP	12350297	firefox
1484673	4	1325445237825884	d1e0229	12170293	58104	993	TCP	178a02f1	thebat
3429674	4	1325445236820017	d1e0229	14cb02b9	61159	80	TCP	178a02f1	Dropbox
3329860	1	1325445237044777	d20022b	1199028a	47801	80	TCP	12350297	plugin-container
3829589	1	1325445236797638	d20022b	124d0296	36868	80	TCP	12350297	wget
3474027	4	1325445212663601	d1e0229	14db02c2	63409	24536	UDP	178a02f1	Skype
4194793	1	1325445280781252	d20022b	1206028f	53331	22849	TCP	12350297	amule

Table II
ONE TCP COMMUNICATION STORED IN THE DATABASE

flow id	direction	packet size [B]	SYN flag	ACK flag	PSH flag	FIN flag	RST flag	CWR flag	ECN flag	URG flag	relative timestamp [μs]	content type
2784673	OUT	60	1	0	0	0	0	0	0	0	0	1
2784673	IN	60	1	1	0	0	0	0	0	0	30012	1
2784673	OUT	52	0	1	0	0	0	0	0	0	44	1
2784673	OUT	431	0	1	1	0	0	0	0	0	395	1
2784673	IN	52	0	1	0	0	0	0	0	0	30241	1
2784673	IN	527	0	1	1	0	0	0	0	0	2554	22
2784673	OUT	52	0	1	0	0	0	0	0	0	27	1
2784673	IN	539	0	1	1	0	0	0	0	0	10455	22
2784673	OUT	52	0	1	0	0	0	0	0	0	15	1
2784673	OUT	287	0	1	1	0	0	0	0	0	1383	1
2784673	OUT	52	0	1	0	1	0	0	0	0	15047	1
2784673	IN	269	0	1	1	0	0	0	0	0	16408	1
2784673	OUT	40	0	0	0	0	1	0	0	0	45	1
2784673	IN	52	0	1	0	1	0	0	0	0	13354	1
2784673	OUT	40	0	0	0	0	1	0	0	0	29	1

approach, focusing on recognizing different kinds of HTTP content (audio, video, file downloads, interactive websites) was presented in [21].

VI. OBTAINING PER-APPLICATION STATISTICS

The next step was to obtain statistical profiles of flows for different applications. Therefore, we developed a tool for calculating statistics on several traffic attributes for each flow in the database, which fulfills our requirements. The statistics include 32 attributes based on sizes and 10 protocol-dependent attributes [20]. We suspect that the attributes based on sizes are independent of the current conditions in the network (like for example congestion). All the protocol-dependent attributes are very general. Precise port numbers are not used, but only information about whether the port is well-known or dynamic. This way we avoid constructing a port-based classifier, but we can retain the information if the application model is more like client-server or peer-to-peer.

The general calculated statistics are [20]:

- number of inbound / outbound / total payload bytes in the sample.
- proportion of inbound to outbound data packets / payload

bytes.

- mean, minimum, maximum first quartile, median, third quartile, standard deviation of inbound / outbound / total payload size in the probe.
- ratio of small inbound data packets containing 50 B payload or less to all inbound data packets.
- ratio of small outbound data packets containing 50 B payload or less to all outbound data packets.
- ratio of all small data packets containing 50 B payload or less to all data packets.
- ratio of large inbound data packets containing 1300 B payload or more to all inbound data packets.
- ratio of large outbound data packets containing 1300 B payload or more to all outbound data packets.
- ratio of all large data packets containing 1300 B payload or more to all data packets.
- application: skype, ftp, torrent, web, web_radio, game, ssh.

The protocol-dependent attributes are [20]:

- transport protocol: TCP, UDP.
- local port: well-known, dynamic.
- remote port: well-known, dynamic.
- number of ACK / PSH flags for the inbound / outbound direction: continuous.
- proportion of inbound packets without payload to inbound packets: continuous.
- proportion of outbound packets without payload to outbound packets: continuous.
- proportion of packets without payload to all the packets: continuous.

The precise process of obtaining these statistics was described in detail and evaluated in [20]:

VII. Machine Learning Algorithms

In the recent literature we can find numerous approaches to use Machine Learning Algorithms to classify the traffic in computer networks. The most widely used MLA classifiers are C4.5 [9] and its modified Java implementation called J48 [12], [22]. Based on statistical analysis, MLAs have the ability to assign a particular class (like P2P) even to traffic generated by unknown applications [9]. It was also proved in [22] that the statistical parameters for encrypted and unencrypted traffic produced by the same application are similar and, therefore, the encrypted payload does not influence results of the training or the classification. The accuracy of the classification by MLAs was claimed to be over 95 % [9]–[11], [13], [14], [23]–[25]. The analysis of the related work can be found in [20].

It was found in [11] that results of the classification are most accurate when the classifier was trained in the same network as the classification process was performed. This may be due to different parameters, which are constant in the particular network, but which differ among various networks. A good example is the Maximum Transmission Unit, which can easily influence statistics based on sizes. Therefore, in our design we decided to train the classifier by volunteers in the same network as the classifier will be installed. This allows us to make a self-learning system, where a group of volunteers

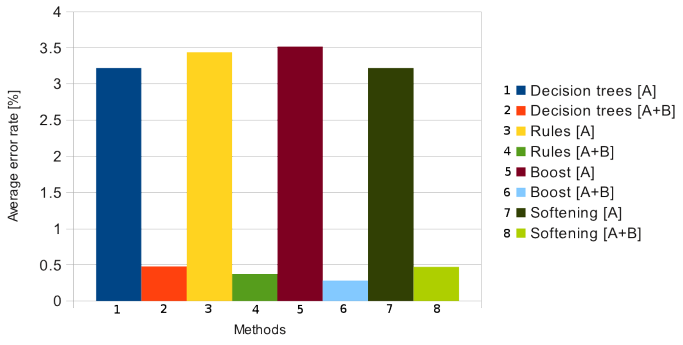

Figure 3. Average error rates of the classifiers [20]

in the network deliver data used for training the classifier constantly improving its accuracy, while all the users can be monitored in the core using the generated decision rules. The next advantage of the design is that even if some network users cannot participate in the data collecting process because of using other operating systems or devices than supported (like MacOS, Apple or Android smartphones), they will still be able to be monitored in the core of the network because of rules created on the basis of data collected from the other users.

Our system uses the C5.0 MLA, which is a successor of C4.5. It is proven to have many advantages over its predecessor, such as higher accuracy, possibilities to use boosting, pruning, weighting and winnowing attributes. Furthermore, the time to generate the decision tree or rules drastically decreased [26]. In order to test the efficiency of C5.0, we performed a set of tests during which we used various training and classification options. The training statistics were obtained from the data provided by our VBS. During our research we found relevant set of arguments and discovered that the best results were obtained using the boosted classifier. The average accuracy fluctuated between 99.3 % and 99.9 % depending on number of training and test cases and amount of data from each case. This behavior is illustrated in Figure 3. It is worth mentioning that in our experiment we considered only 7 different groups of applications and only flows longer than 15 packets. In our small-scale prototype for tests we decided to limit the number of applications and take into account Skype, FTP, torrent, web traffic, web radio traffic, interactive game traffic and SSH [20]. The limitation of the flow length was done because we needed to have at least 5 packets to generate the statistics (the first 10 packets of each flow were skipped as their behavior is different than the behavior of the rest of the flow). The detailed description of our methods and results can be found in [20]. The decision tree generated in this step can be used to classify the traffic in the real network.

VIII. The centralized monitoring solution

This paragraph presents the proposed design of the centralized monitoring solution which can be placed in any point in the network to examine network QoS.

Because of heavy load in the high-speed networks, it is not possible to monitor all the flows passing the central point at the same time. Therefore, statistics from only selected flows can be captured and passed to the C5.0. Selection of such flows can be based on two methods: Capturing one flow per user and intelligent switching between the flows. From the QoS point of view, it is important to discover problems with a particular user or to inform the user that problems experienced by him are results of problems in the remote network. If it is the user who has the problem, then the problem usually influences all the user's network activity.

Each application has some special requirements regarding network parameters. When a small congestion occurs, the service level can still be sufficient for P2P file downloads, but Skype communication may be not possible because of big jitter and delays. It is, therefore, not sufficient to monitor one random flow at a time, but we need to monitor a flow which have high quality requirements. Our solution should be built based on the following assumptions:

- Only one flow per user at a time is consistently monitored for QoS.
- Statistics for another random flow per user at a time are passed to C5.0 to discover the application.
- If the application has higher QoS requirements than the currently monitored, switch monitoring to the new flow; if not, stick to the current.
- If monitoring of the selected flow discovers problems, start monitoring few flows at a time to check if this problem lay on the user's side or on the remote side.

Because of the dynamic switching between the flows when determining the application, it is most probable that the system will not be able to capture flows from their beginning. The classifier designed by us, which use the C5.0, is able to determine the application on the basis of the given number of packets from any point in the flow [20].

Monitoring of the QoS can be done in passive or active mode. The passive mode relies mostly on time-based statistics, which are obtained directly from the flow passing the measurement point. This way, we can assess the jitter, the burstiness and the transmission speed (both download and upload). Unfortunately, it is not possible to receive information about the packet loss or the delay for other than TCP streams while using this method. For that reason, additional tools performing active measurements must be involved in the process of estimating the QoS. One option is to use the ping-based approach, as it can measure both delay and packet loss. Unfortunately, other issues can arise. Ping requests and responses are often blocked by network administrator, or their priority is modified (decreased to save the bandwidth or increased to cheat the users about the quality of the connection). Other options include sending IP packets with various TTL and awaiting *Time Exceeded* ICMP messages, which are usually allowed to be transmitted in all the networks and their priority is not changed. Active measurements must be done in both directions (from the user and from the remote side). The total packet loss and the delay can be calculated as the sum of the delays and the packet losses from both directions of the flow. Furthermore, the knowledge of the direction that causes problems can be used to assess if the problems are located in the local network or somewhere outside.

IX. Conclusion

The paper shows a novel method for assessing the Quality of Service in computer networks. Our approach involves a group of volunteers from the target network to participate in the initial training of the system, and later in the self-learning process. The accurate data obtained from the volunteers are used by the C5.0 MLA to create the per-application profiles of the network traffic as classification decision trees. The centralized measurement system uses the decision trees to determine the applications associated with flows passing through the measurement point. This knowledge allows us to precisely define the QoS requirements for each particular flow. To assess the QoS level two methods are proposed: The passive and the active one.

X. Acknowledgment

This research work was conducted in the context of a PhD project in the Section for Networking and Security in the Department of Electronic Systems at Aalborg University. We are also very grateful, for the possibility to install and test our Volunteer-Based System, to *Gimnazjum nr 3 z Oddziałami Integracyjnymi i Dwujęzycznymi imienia Karola Wojtyły w Mysłowicach*, a high school in Poland.

References

[1] Tomasz Bujlow, Tahir Riaz, Jens Myrup Pedersen, *A Method for Assessing Quality of Service in Broadband Networks*, Proceedings of the 14th International Conference on Advanced Communication Technology (ICACT 2012), IEEE 2012, pp. 826–831.

[2] Gerhard Haßlinger, *Implications of Traffic Characteristics on Quality of Service in Broadband Multi Service Networks*, Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04), IEEE Computer Society 2004.

[3] Thomas M. Chen, *Internet Performance Monitoring*, Proceedings of the IEEE, vol. 90, no. 9, September 2002, pp. 1592–1603.

[4] LIRNEasia Broadband QoSE Benchmarking project, 2008. [Online]. available: http://lirneasia.net/projects/2008-2010/indicators-continued/broadband-benchmarking-qos-20/

[5] K. v. M. Naidu, Rajeev Rastogi, Bell Labs Research India, Bangalore, *Detecting Anomalies Using End-to-End Path Measurements*, IEEE INFOCOM 2008 proceedings, 2008, pp. 16–20.

[6] A. Dogman, R. Saatchi, S. Al-Khayatt, *An Adaptive Statistical Sampling Technique for Computer Network Traffic*, IEEE CSNDSP 2010, pp. 479–483.

[7] Baek-Young Choi, Jaesung Park, Zhi-Li Zhang, *Adaptive Random Sampling for Traffic Load Measurement*, IEEE International Conference on Communications, IEEE 2003, pp. 1552–1556.

[8] Shao Liu, Mung Chiang, Mathias Jourdain, Jin Li, *Congestion Location Detection: Methodology, Algorithm, and Performance*, 17th International Workshop on Quality of Service, IEEE 2009.

[9] Jun Li, Shunyi Zhang, Yanqing Lu, Junrong Yan, *Real-time P2P traffic identification*, IEEE GLOBECOM 2008 PROCEEDINGS.

[10] Riyad Alshammari, A. Nur Zincir-Heywood, *Machine Learning based encrypted traffic classification: identifying SSH and Skype*, Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009).

[11] Sven Ubik, Petr Žejdl, *Evaluating application-layer classification using a Machine Learning technique over different high speed networks*, 2010 Fifth International Conference on Systems and Networks Communications, IEEE 2010, pp. 387–391.

[12] Ying Zhang, Hongbo Wang, Shiduan Cheng, *A Method for Real-Time Peer-to-Peer Traffic Classification Based on C4.5*, 12th IEEE International Conference on Communication Technology, IEEE 2010, pp. 1192–1195.

[13] Jing Cai, Zhibin Zhang, Xinbo Song, *An analysis of UDP traffic classification*, 12th IEEE International Conference on Communication Technology, IEEE 2010, pp. 116–119.

[14] Riyad Alshammari, A. Nur Zincir-Heywood, *Unveiling Skype encrypted tunnels using GP*, IEEE Congress on Evolutionary Computation (CEC), IEEE 2010.

[15] Kartheepan Balachandran, Jacob Honoré Broberg, Kasper Revsbech, Jens Myrup Pedersen, *Volunteer-based distributed traffic data collection system*, Feb. 7-10, 2010 ICACT 2010, pp. 1147–1152.

[16] Kartheepan Balachandran, Jacob Honoré Broberg, *Volunteer-based distributed traffic data collection system*, Master Thesis at Aalborg University, Department of Electronic Systems, June 2010.

[17] Tomasz Bujlow, Kartheepan Balachandran, Tahir Riaz, Jens Myrup Pedersen, *Volunteer-Based System for classification of traffic in computer networks*, 19th Telecommunications Forum TELFOR 2011, IEEE 2011, pp. 210–213.

[18] Tomasz Bujlow, Kartheepan Balachandran, Sara Ligaard Nørgaard Hald, Tahir Riaz, Jens Myrup Pedersen, *Volunteer-Based System for research on the Internet traffic*, to appear in Telfor Journal Vol. 4 (2011).

[19] Volunteer-Based System for Research on the Internet, 2012. [Online]. Available: http://vbsi.sourceforge.net/

[20] Tomasz Bujlow, Tahir Riaz, Jens Myrup Pedersen, *A method for classification of network traffic based on C5.0 Machine Learning Algorithm*, International Conference on Computing, Networking and Communications (ICNC 2012), IEEE 2012, pp. 244–248.

[21] Tomasz Bujlow, Tahir Riaz, Jens Myrup Pedersen, *Classification of HTTP traffic based on C5.0 Machine Learning Algorithm*, to appear in Fourth IEEE International Workshop on Performance Evaluation of Communications in Distributed Systems and Wed based Service Architectures (PEDISWESA 2012).

[22] Jason But, Philip Branch, Tung Le, *Rapid identification of BitTorrent Traffic*, 35th Annual IEEE Conference on Local Computer Networks, IEEE 2010, pp. 536–543.

[23] Jun Li, Shunyi Zhang, Yanqing Lu, Zailong Zhang, *Internet Traffic Classification Using Machine Learning*, Second International Conference on Communications and Networking in China, CHINACOM '07, 2007, pp. 239–243.

[24] Yongli Ma, Zongjue Qian, Guochu Shou, Yihong Hu, *Study of Information Network Traffic Identification Based on C4.5 Algorithm*, 4th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE 2008.

[25] Wei Li, Andrew W. Moore, *A Machine Learning Approach for Efficient Traffic Classification*, Proceedings of the Fifteenth IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'07), IEEE 2008, pp. 310–317.

[26] Is See5/C5.0 Better Than C4.5?, 2009. [Online]. Available: http://www.rulequest.com/see5-comparison.html

Tomasz Bujlow is working as a Ph.D. Student in the Section for Networking and Security (NetSec) in the Department of Electronic Systems at Aalborg University in Denmark. He received his Master of Science in Computer Engineering from Silesian University of Technology in Poland in 2008, specializing in Databases, Computer Networks and Computer Systems. Previously, he obtained his Bachelor of Computer Engineering from University of Southern Denmark in 2009, specializing in software engineering and system integration. His research interests include methods for measurement of Quality of Service and traffic classification in computer networks. He is also a Cisco Certified Network Professional (CCNP) since 2010.

Sara Ligaard Nørgaard Hald is working as a Ph.D. student in the Section for Networking and Security (NetSec) in the Department of Electronic Systems at Aalborg University in Denmark. She received her Master of Science in Computer Engineering and Management from Aalborg University in 2002, and has since worked for the Danish Defense and as a consultant specializing in enterprise architecture and cybersecurity. Research interests include threat assessments and attack detection in dedicated networks.

Tahir Riaz is working as an Assistant Professor in the Section for Networking and Security (NetSec) in the Department of Electronic Systems at Aalborg University in Denmark. He received his Master and PhD degrees in Electrical and Electronics Engineering, specializing in Network Planning and Management, from Aalborg University in 2004 and 2008, respectively. He has also worked in Nokia, Linkoping, Sweden. He has authored or co-authored over 70 papers published in conferences and journals. His research interests include access and backbone fiber optic networks, network planning and design, architecture of next generation radio of fiber networks, reliability and QoS issues in large-scale access and core network infrastructures, performance and optimization in networks.

Jens Myrup Pedersen is working as an Associate Professor and the head of the Section for Networking and Security (NetSec) in the Department of Electronic Systems at Aalborg University. His current research interests include network planning, traffic monitoring, and network security. He obtained his Master of Science in Mathematics and Computer Science from Aalborg University in 2002, and his PhD in Electrical Engineering also from Aalborg University in 2005. He is an author/co-author of more than 70 publications in international conferences and journals, and has participated in Danish, Nordic and European funded research projects. He is also a board member of a number of companies within technology and innovation.