BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
DEPT. OF TELECOMMUNICATIONS AND MEDIA INFORMATICS

METHODS FOR EFFICIENT CLASSIFICATION OF NETWORK TRAFFIC

Géza Szabó

Summary of the Ph.D. Dissertation

Supervised by

Dr. Sándor Molnár PhD
High Speed Networks Laboratory
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics

Budapest, Hungary
2010

1 Introduction

Detailed knowledge about the traffic mixture is essential for network operators and administrators, as it is a key input for numerous network management activities. For example, it can be the input for network planning and capacity provisioning, fine-tuning of charging schemes, or security monitoring. Further, a good understanding of the traffic mix and the capability of observing trends in the traffic volume of the different applications can provide important input to network equipment design.

The aim of traffic classification is to find out what type of applications are run by the end users, and what is the share of the traffic generated by the different applications in the total traffic mix. The communication between IP network nodes can be organized into flows, and the task of traffic classification is to assign a specific application to each individual flow. A flow is a collection of IP packets sent from a given port at one IP address to a given port at another IP address using a specific protocol. A flow is identified by its five-tuple flow identifier: source IP address, destination IP address, source port, destination port, protocol identifier.

The most accurate traffic classification would obviously be complete protocol parsing. However, many protocols are ciphered due to security reasons (SSH [8], SSL [7]). Also some are proprietary, thus there is no public description available (Skype [9], MSN Messenger [5], World of Warcraft [10], etc.). In general, it would be difficult to implement every protocol which can occur in the network. In addition, even simple protocol state tracking with a high number of users can make the method so resource consuming that it becomes practically infeasible.

- **Port based classification:** In the simplest and most common method the classification is based on associating a well-known port number with a given traffic type, e.g., web traffic with TCP port 80 [4].

- **Signature based classification - Deep Packet Inspection (DPI) [40]:** To make protocol recognition feasible, only specific byte patterns are searched in the packets in a stateless manner. These byte signatures are predefined to make it possible to identify particular traffic types, e.g., web traffic contains the string '`GET`', eDonkey peer-to-peer (P2P) traffic contains '`e3 38`'.

- **Traffic characteristics based classification:** In statistics based classification some statistical features of the trace are captured and used to classify the network traffic. To automatically discover the features of a specific kind of traffic, the statistical methods are combined with methods coming from the field of artificial intelligence. The most frequently discussed method is the Bayesian analysis technique as in [36], [35], [22], [15].

- **Connection pattern based classification [29],[30]:** The basic idea is to look at the communication pattern generated by a particular host, and to compare it to the behavior patterns representing different activities/applications.

- **Information theory based classification:** A useful aid in traffic classification is introduced in [46] and [32] which is an information theoretic approach and can group the hosts into typical behaviors e.g., servers, attackers. The main idea is to look at the variability or randomness of the set of values that appear in the five-tuple of the flow identifiers, which belong to a particular source or destination IP address, source or destination port. Practically this method is a good quantification of the cardinality of the graph nodes of the connection pattern based method.

2 Research Objectives

The objective of the dissertation is to design and develop robust and efficient traffic classification tools for IP networks. My **first goal** was to establish a framework to make it possible to compare the traffic classification mechanisms available in literature. I proposed a method for traffic classification validation. The further goal was to utilize the results of the measurement of classification metrics to propose a new combined method for traffic classification.

During the validation of traffic classification methods, it turned out that the presumptions used in traffic characteristics based classifiers in current literature is obsolete when the identification of currently popular gaming traffic is needed. Most of current internet traffic consists of proprietary RTS and MMORPG traffic beside the former modeled FPS games. FPS traffic consists of fixed sized packets sent in fixed rate. Current popular gaming traffic is special as the traffic characteristics are highly influenced by user interactions. In the **second** part the **goal** was to analyze the traffic characteristics of these new emerging game types.

Gaming traffic usually uses proprietary non-documented protocols, further even Protocol Header Encryption to make protocol recognition unfeasible with common DPI methods. During my work with gaming traffic, I experienced that in most cases gaming traffic consists of the communication of player related coordinates between peers. I analyzed the possibility of the utilization of this information. In the **third** part the **goal** was to extend current DPI methods to make them also capable of recognizing fields with non-fixed byte signatures in a protocol structure.

The first part of my work focused on traffic classification accuracy and completeness and not considered the traffic classification speed. However, in most cases this

is the go not go decision whether a method can be applied in a real-time traffic classification system. In the **fourth** part the **goal** was to focus on wire-speed traffic classification and to reformulate traffic classification parts to utilize efficiently massively parallel architectures.

3 Methodology

In Thesis 1,4, the proposed methods were *implemented* and tested in real operational networks or *measurements* obtained from them.

Results reported in Thesis 2 were obtained by *statistical analysis* of active measurements. In Thesis 3 the model parameters were obtained through the statistical analysis of measured data from real networks.

In Thesis 3 the analysis of the effects of process decomposition and in Thesis 4 the achievable compression ratio were analyzed through custom-built *simulations* implemented in Matlab [11].

4 New Results

4.1 Novel framework for traffic classification and validation

First, I deal with the problem of the validation of traffic classification algorithms. Several different classification approaches can be found in the literature. However, the validation of these methods is weak and ad hoc, because neither a reliable and widely accepted validation technique nor reference packet traces with well-defined content are available. The validation is typically done with another specific classification method.

Sen et al. [40] validated their constructed signature database by manually checking the false positive ratio of their technique. Karagiannis et al. [30] validated their method by using signature based classification. As there are no commonly accepted and well performing byte-signatures, authors constructed their own signature database. Crotti et al. [19] authors used port based classification method while in [23] used payload based classification method to create training and evaluation data set from the measurements collected at their campus network. Erman et al. [22] worked with commonly available traffic traces, but these traces contained only packet headers which excludes such reliable validation methods which are based on packet payload. In [15], the traffic classification method was applied online without capturing the original data due to the lack of capacity to store the massive amount of data

which is the consequence of high traffic speeds. This makes impossible to validate the traffic classification by others.

Thesis 1. *[J1, C1, C2, C3] I introduce a new validation technique for traffic classification methods. I also present a novel combined traffic classification method. [Chapter 2 of dissertation]*

4.1.1 Validation of traffic classification mehods

A classification method can be convincingly validated only by an active test, for which a number of requirements are fulfilled, such as:

- It should be independent from classification methods, i.e., the validation of a classification method by another one must be avoided,
- About each packet the test should provide reference information that can be compared to the result of the classification method under study,
- The test should be deterministic, meaning that it should not rely on any probabilistic decisions,
- Feasibility: it should be possible to create large tests in a highly automated way, and
- The environment where the active measurements are collected should be realistic.

Thesis 1.1. *[C3] I propose a method for validation of traffic classification by extending the functionality of the traffic generating terminals. The method adds an application level information to the packets of the generating applications at transport layer.*

The two main requirements on the realization of the method are that it should not deteriorate the performance of the terminal, and the byte overhead of marking should also be negligible. The preferred realization is a driver that can easily be installed on terminals. The position of the introduced driver can be seen in Figure 1. It takes place right before the network interface thus each packet exchanged between the terminal and the network has to pass through it. I have implemented a prototype, which is a Windows XP driver based on the Network Driver Interface Specification (NDIS) library. The kernel NDIS library abstracts the network hardware from network drivers and provides an API through which intelligent network drivers can be efficiently programmed. If the sending and receiving functions of the NDIS IP protocol driver are hooked, all TCP and UDP packets can be intercepted and filtered.

To meet our requirements the driver is designed to work in the following way. In the case of a passing through packet the following process takes place (see Figure 2):

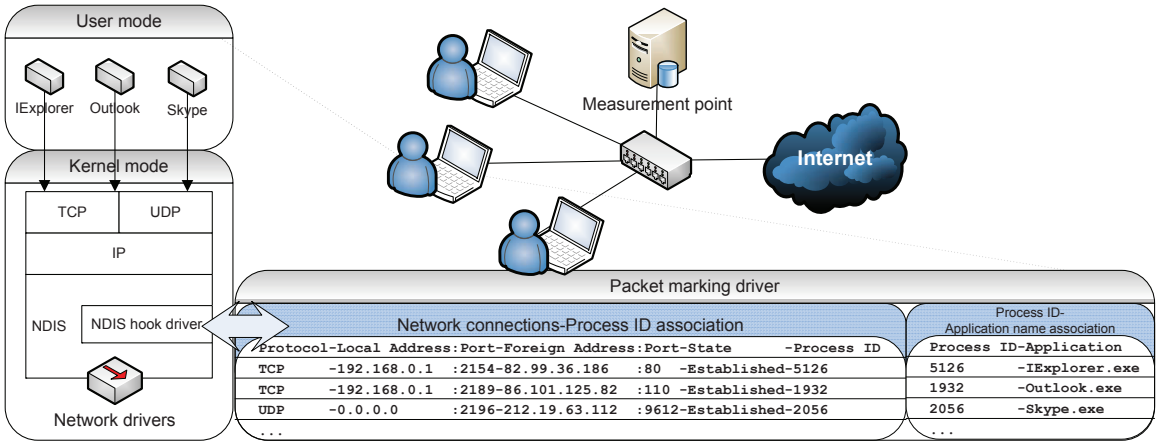


Figure 1: The position of the proposed driver within the terminal

1. The packet is examined whether it is an incoming or outgoing packet. In case of an incoming packet the process ends without marking the packet as it is not beneficial to mark incoming packets.

2. In case of an outgoing packet the size of the packet is examined. If the current packet size is already the size of Maximum Transmission Unit (MTU), the extension of the packet with marking would lead to IP fragmentation. To avoid this the process continues with only those packets which are smaller than the MTU decreased with the size of marking.

3. As there is no information in the operating system about those 'network connections' which use other protocols than TCP or UDP, the process continues with only TCP or UDP packets.

4. According to the five-tuple identifier of the packet it is checked whether there is already available information about which application the flow belongs to. If there is no information on the flow in the cache yet, the operating system is queried to supply the established network connections and the process IDs of the responsible applications.

5. When all information is prepared for the marking of the packet, there is a final chance to decide whether the driver should mark the packet or not. The packet marking can be done for all of the packets in the flow, randomly selected packets of the flow, only the first packet of the flow or it is also possible to switch off the marking for specific applications.

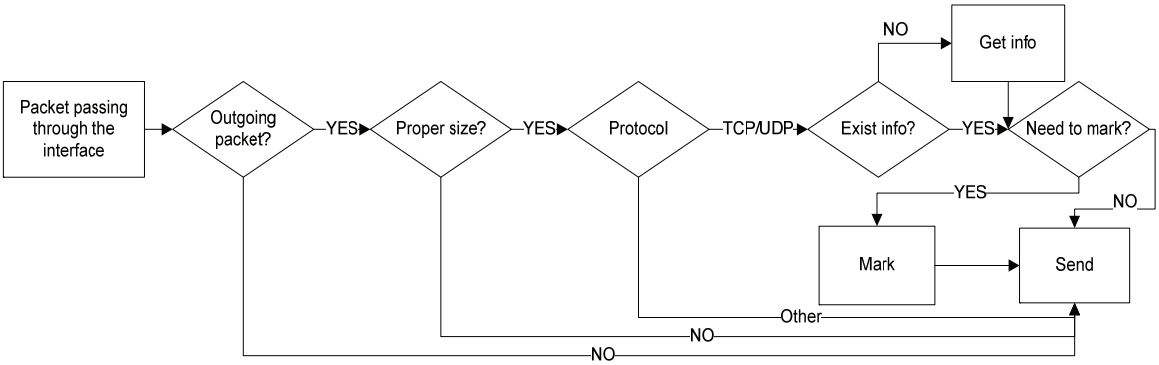


Figure 2: The working mechanism of the introduced driver

Evaluation. The validation was done by using the method in real network conditions by installing it onto several client machines accessing the internet. The method worked transparently for the communication parties, however, this was achieved by several additional functionalities extending the basic idea (for details see [C3]).

The requirements are fulfilled by the proposed method:

- It should be independent from classification methods: This is fulfilled as traffic classification is done during the traffic generation instead of postprocessing an unknown traffic;
- About each packet the test should provide reference information: Traffic marking is done on packet level, even each packet can be marked optionally;
- The test should be deterministic: The system is composed only deterministic building blocks;
- Feasibility: The only manual work is the installation of the system, thus the system is highly automated;
- The environment where the active measurements are collected should be realistic: The system does not introduce any constraint to the users' everyday behavior thus installing the system to real-life users the collected measurements are realistic.

The two key metrics that can characterize the performance of a traffic classification method are accuracy and completeness [30]:

- *Accuracy* means the fraction of traffic identified correctly in the sense that it is associated to the application type which indeed generated it. The classification accuracy values of an examined method show the ratio of the correctly classified

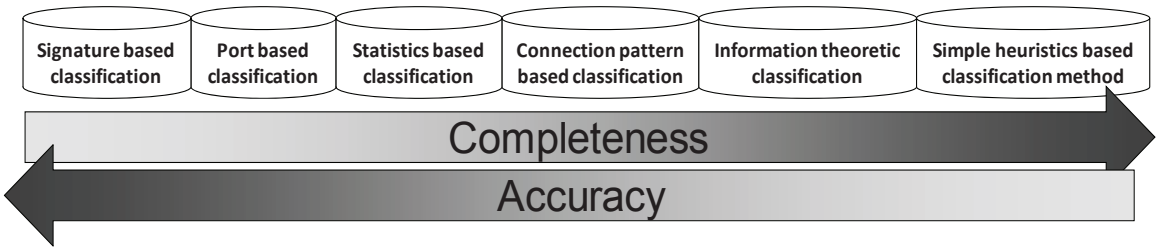


Figure 3: The accuracy and completeness rank of different traffic classification methods

traffic compared to all the traffic classified by the examined method for the given application. This can be calculated by the following method: $\mathcal{A} = \frac{TP}{TP+FP}$, where TP and FP are representing the true positive and false positive decisions for a flow considering an examined method.

- *Completeness* means the fraction of traffic which is categorized by a particular method. It is the fraction of the traffic which the method is capable of associating to one of the application types. The classification ratio values shows the ratio of flows classified as a specific application type by the examined method and the application marked in the validation trace. This can be calculated by the following method: $\mathcal{C} = \frac{TP+FP}{TP+FN}$ with the same denotations as in $\mathcal{A}$.

The perfect classification is the $(\mathcal{A}, \mathcal{C}) = (1, 1)$ case. During the classification both of these metrics are intended to be kept around the perfect classification values. The following example shows the accuraccy and completeness of two applications classified with one specific method (e.g., `A`) comparing to the reference trace (e.g., `A*`).

```
A - A*, A - A*, A - B*, B - B*, ? - B*
Accuracy: A: 2/3 B: 1/1, Completeness: A: 3/2 B: 1/3
```

The experienced accuracy and completeness about the traffic classification methods is depicted in Figure 3. The most accurate methods are the strict protocol parsers and their lightweight implementation, the payload based traffic classifier. The most complete solutions are those methods which rely mostly on weak heuristics. In this way, such methods can always classify the traffic thus their completeness will be high, however their accuracy is low. It should be noted that Figure 3 is a general overview of the accuracy and completeness of current methods but it is not intended to mean that the trade-off between these metrics always exist. For the exact accuracy and completeness metrics referring to several application types see [C1] for details.

4.1.2 Combined traffic classification method

Before this work, papers searched for one ultimate solution to cope with the traffic classification problem in general. I proved that traffic classification methods vary in their accuracy per application types. A complex decision mechanism is needed to provide high accuracy and completeness in all real-world situations.

To combine existing traffic classification mechanisms, decision trees were chosen to exploit their following advantages: a) decision trees can be implemented and used efficiently, b) decision trees are simple to understand and interpret. The machine learning technique for inducing a decision tree from data is called decision tree learning. During decision tree construction algorithms for discrete data sets, the cost function is generally information gain related. This metric is not directly related to the traffic classification task. Information gain related metrics neglect application types having less flows than others e.g., gaming traffic comparing to P2P.

Thesis 1.2. *[J1, C1, C2] I propose a method to select the adequate traffic classification engines for a specific application type. The method incorporates a decision mechanism and heuristics based on the measured accuracy of the classification methods. An application of the concept is incorporated into an automatic decision tree construction mechanism. The cost function of the decision tree optimization algorithm is customized for traffic classification.*

I propose to optimize on the traffic classification metrics directly during decision tree construction. My proposal is to handle all applications and traffic classification methods with equal weight. To optimize on both accuracy and completeness I define the target function as the Euclidean-distance from the perfect classification $(\mathcal{A}, \mathcal{C}) = (1, 1)$ of the calculated average accuracy and completeness of a specific traffic classification method $m \in M$ on a set of flows F containing application set A.

$$f(F, M, A) = \min_{m_i \in M} \{((1,1) - \frac{1}{|A|} \sum_{a \in A} (\mathcal{A}^a_{m_1}(F), \mathcal{C}^a_{m_1}(F))), ((1,1) - \frac{1}{|A|} \sum_{a \in A} (\mathcal{A}^a_{m_2}(F), \mathcal{C}^a_{m_2}(F))), ...\}$$

, with the same denomination as in Section 4.1.1.

Evaluation. The method is applicable for analysis of traffic mixture of network traces. Due to the speed limitation of some applied algorithms the traffic classification can be done mainly offline.

I validated the combined traffic classification method with the proposed validation mechanism in Thesis 1.1, with the addition that the classification of VoIP applications has been extended with ideas from [37]. Figure 4 shows the improvement of both accuracy and completeness of the introduced combined classification comparing to the

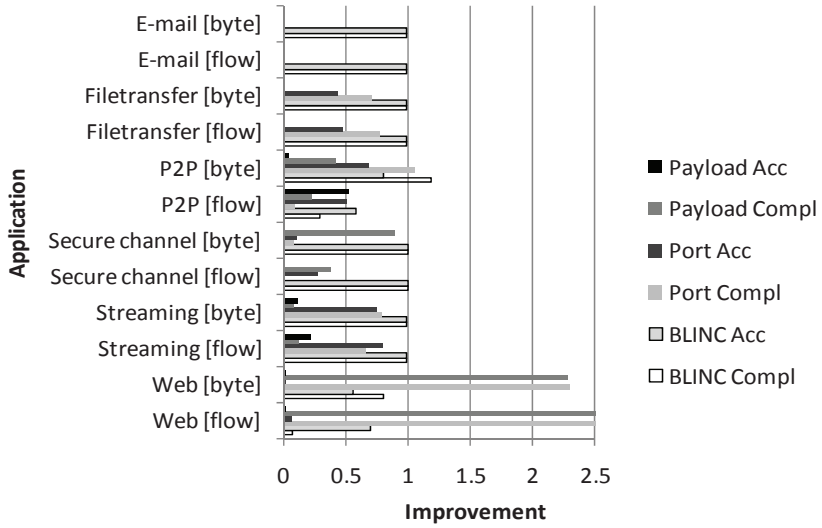


Figure 4: The improvement of accuracy and completeness of the introduced combined classification comparing to the individual methods

individual methods [C1]. The improvement of both of the measures can be observed in all application types. For details see [C3].

4.2 Characteristics of gaming traffic

Gaming traffic analysis is important for traffic classification due to its emerging penetration, and for simulator supported gaming server architecture design.

Game traffic depends on two main factors, the game protocol and the gamers' behavior. Based on popular real-time multiplayer games in this thesis group I investigated the latter factor showing how a set of typical game phases – e.g., player movement, changes in the environment – impacts traffic on different observation levels. The nature of human behavior has such a high impact on traffic characteristics that it influences the traffic both at a macroscopic – e.g., traffic rate – and at a microscopic – payload content – level. The role of player activity can be important for modeling status update message intensity coming from game server while environment changes can imply server exchange.

Fernandes et al. [24] the authors showed what happens when an avatar performs different actions in the virtual world, at different places and under different network conditions. My work extends the investigation of gaming traffic of [24] in the function of time gaining information about gaming environments, further I redefine the player activity applied in [18] to gain information about player behavior. My work aims at deriving detailed character movement and gaming environment information (see

e.g., [24]) but obtaining this information from non-intrusive measurements without the requirement of the detailed knowledge of the whole protocol as in e.g., [41] or [34].

Emerging Massively Multiplayer Online Real Time Strategy games require complex game server architecture to make the transmission of the state information of a huge number of units generated by a lot of players feasible. This architecture design is supported by network traffic simulations based on the accurate characterization of player behavior [47]. However, player behavior characteristics in RTS games influencing network traffic have not been investigated yet, thus, this is the main motivation of this work. In particular, I introduce a method that can identify the war periods in real-time strategy game sessions based on non-intrusive measurements thus, it is possible to analyze a vast number of game plays.

Thesis 2. *[J2, J4, J5, C4, C5] I propose methods to identify several gaming actions in Massively Multiplayer Online Role Playing Games (MMORPG) and Real-Time Strategy (RTS) games. [Chapter 3 of dissertation]*

Before my work there were no papers dealing with player behavior and gaming environment analysis of MMORPG traffic based on non-intrusive measurements. Other works either modify the game itself, turn on some client side logging which is later collected or reverse engineer the protocol messages and parse the packets. My traffic characteristics based method can be applied for a wide range of games as it is protocol format independent, it builds on the common traffic characteristics of this game genre.

4.2.1 User behavior detection algorithm for MMORPG traffic

Thesis 2.1. *[J2, J4, J5, C4] I propose a method to identify player behavior and changes in gaming environment in MMORPGs. The method is based on wavelet decomposition of the traffic rate and to track the environment changes and player behavior with a state-machine which state transitions are dependent on the wavelet components. The high frequency component change makes a state transition between stalling and moving, while the low frequency component change results in a state transition between the environments. The method works on the category of MMORPGs which are sending gaming environment updates to the player around the player's certain proximity. This covers all known MMORPGs.*

In Figure 6 the wavelet decomposition of the traffic presented in Figure 5 can be seen. It can be noted that the moving and stalling states are well-distinguishable on the high-frequency component: in case of moving, it becomes noisier. The environment changes can be well-tracked on the low-frequency component. The state-machine tracks whether the player is inside city or outside city and does not let

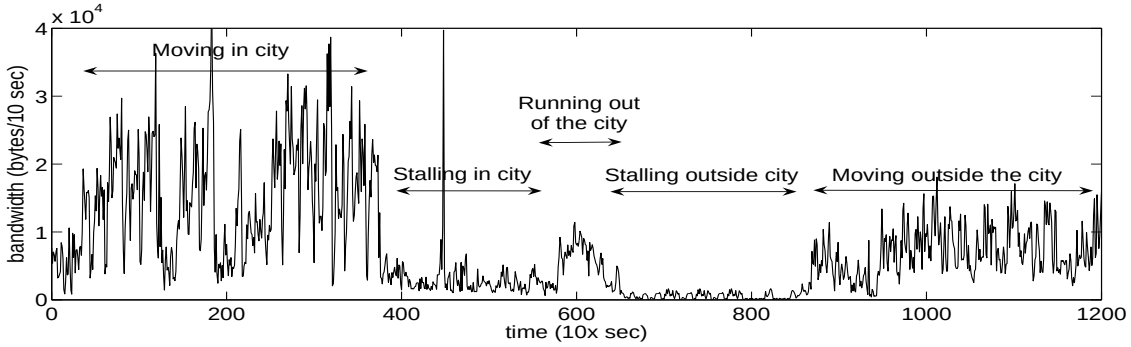


Figure 5: World of Warcraft measurement bandwidth (bytes/10 sec)

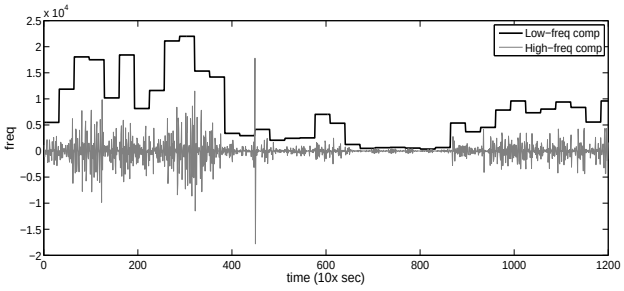


Figure 6: Analysis of World of Warcraft traffic and the effect of different environments and character actions

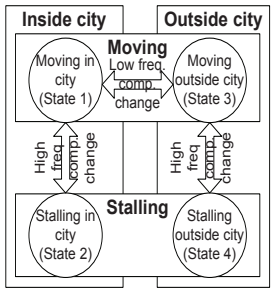


Figure 7: States and defined state transmissions in the proposed method

sudden changes in the environment if temporal transients occur in either of the frequency components. Figure 7 shows the defined states and state transmissions in the proposed detection algorithm.

The algorithm was constructed to adapt to the analyzed traffic, thus, no fix thresholds regarding the typical bandwidth have to be set. The advantage of this feature is that any MMORPG game traffic can be analyzed without previous parameter tuning. It is important to note that the method is independent of the packet directions.

Evaluation. The method can provide network simulators with player behavior parameters. The Long-Range Dependence (LRD) property of the gaming traffic was revealed and it was also showed that the popular explanation for the presence of LRD by heavy-tailed player behavior periods [18] cannot be held.

In the first step of the validation process I created several active measurements in a controlled environment by capturing the generated network traffic of the game at the client side and manually log the time of the different states of the player. After the

measurement, the proposed method was applied on the network traces, and the states which were indicated by the proposed method and the logged states were compared. A state is the estimated action of the character in a given environment in a 10 sec period. In this phase of the validation 74% of the states were exactly recognized and 14% of the states were missed with a nearby state. Nearby state means that the state can be reached within one state transition.

Second, I focused on Silkroad Online and World of Warcraft to check the accuracy of the method thoroughly and in bulk measurement in an automated way via processing of ingame video and the correlation to network traffic. The heuristic algorithm grabs the characteristics of the ingame screens of both the environment and the moving activity. In this phase of the validation 68% of the states were identified correctly and 19% of the states were missed by a nearby state. In the first two validation steps I collected and examined 9 hours of MMORPG traffic (World of Warcraft [10] and Silkroad Online [13]).

In the third phase of the validation process, I created several active measurement traces with several MMORPGs to check how generally applicable the method is. It was found that the method performs well in cases where the player population is high and the character actions generate considerable traffic.

4.2.2 Battle event detection in RTS games

Thesis 2.2. *[C5] I propose a method to identify battle events in RTS games. The method is based on a heuristic traffic rate analysis. The method works by searching for a local peak in the traffic rate, then a decreasing trend ending with local minima.*

To grab these specific characteristics of any RTS client side traffic (see Figure 8) I calculated a linear regression in a sliding window for the traffic rate of the server. The indicators of the battle are the following properties (see Figure 9):

- *Is there a slope?* – Local minima in the gradient (p_1): I searched the coefficients of a polynomial $p(x)$ of first degree that fits the data, $p(x(i))$ to $y(i)$, in the least squares sense. $x(i), i = n..n + k$ stands for the examined time interval where n is the start of the examined time interval, and k is the length of the sliding window; $y(i)$ is the traffic rate in the i^{th} time. The result $\vec{p}$ of $p(x) = p_1x + p_2$ is a two-unit-long row vector containing the polynomial coefficients in descending powers.

- *Is there an increase in the rate?* – Local maxima in the mean rate of the traffic in the sliding window.

- *Does the linear fit well?* – Local minima in the variance of the difference between the fitted polynomial and original time series: $\min\{|y(i) - p(x)|\}, i = n..n + k$.

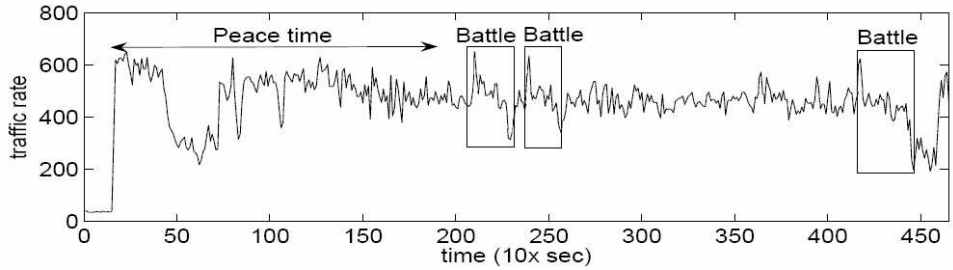


Figure 8: Cossacks measurement traffic intensity (packets/10 sec)

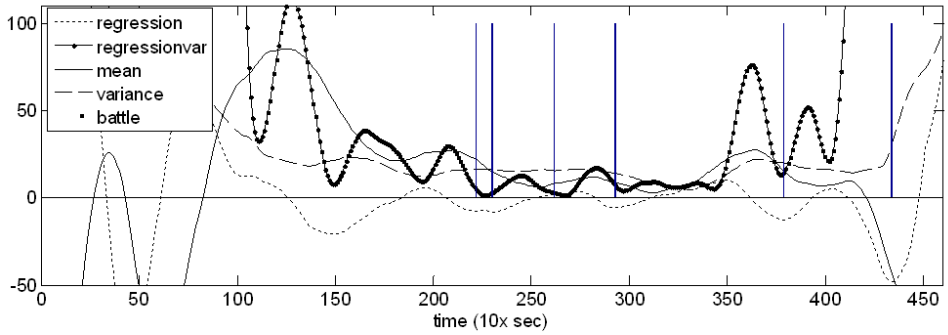


Figure 9: Analysis of Cossacks traffic and the effect of battles during gameplay /battles are indicated with the vertical poles/

- *Is there a unusual jump in the rate?* – The standardized variance of the gradient above a limit: $\tilde{\sigma}\{p_1^i x\} > c, i = n..n + k$.

Evaluation. In the first step I created several active measurements in a controlled environment by capturing the generated network traffic of the game and manually log the time of the battles. After the measurement the proposed method was applied on the network traces. The battles which were indicated by the proposed method and the logged battle events were compared. Overlapping periods of battle periods in the validation and the results of the algorithm are regarded as true positives. In the validation phase 15 hours of RTS gaming traffic were collected and examined In this phase of the validation 33 out of 36 battles (92%) were detected and 3 (8%) of the indicated battles were false positive.

In the second step of the validation process I introduced an automatic method to quantify the size of the battles and to recognize battles automatically not only from the network traffic but from the game itself. This was achieved by the audio processing of the ingame sounds. In this phase of the validation 33 out of 38 battles (87%) were detected and 4 (11%) indicated battles were false positives.

In the third step of the validation process I examined how the proposed method is applicable in general by examining several other RTS games. It was found that the bigger the armies in a given game, the more confident the method is in finding the battles.

The results can be *applied* in MMORTS server design. Providing Real Time Strategy (RTS) game service to a high number of simultaneous players (MMORTS) causes the problem of server load balancing becoming even more complex. The player can manage hundreds of units spread across the virtual world. It is inefficient to transmit all the information around every single unit of the player. Cecin et al. [17] propose to obtain up-to-date data from the server only in those cases when the player is actually concentrating on the specific territory. With the help of my work it is possible to extract territory focusing information for MMORTS player and network simulation.

I also compared the characteristics of the observed RTS games to real world data. I obtained the data about the war activities of the real world from the Correlates of War project [39]. It was found that the distribution of battle durations shows similarities in the gaming and real world environment. I demonstrated that the gaming environment models the real world well but it is not a complete copy of that. This means that parameters obtained from real world can be used to some extent but the most exact results can be obtained by the investigation of the gaming environment.

4.3 Dynamic signature for DPI

Current DPI methods search for fix byte signatures in the packet payloads. However, there are several protocol fields having always changing values resulting in dynamic signatures like timestamps and sequence numbers. This question is even more challenging in case of gaming traffic, where not a single bit or byte changes dynamically, but blocks of bytes describing, e.g., the player movements.

To the best of my knowledge studies focusing on the structure of the packets of the game protocols do not exist yet. Tan et al. [41] and Liang et al. [34] extracted the structure of the gaming protocol from the source code of the game [41] or manually with several empirical experiments [34]. In [38] Pittman et al. used a scripting language [12] which is supported by World of Warcraft to automate ingame activities and extracted the positions of every character they encountered during game. The most related papers in literature focus on automatic signature generation with fix protocol signature subsequences e.g., [25], [31].

I examine how the effects of human behavior can be tracked in network traffic at packet level. Player behavior and movement related packets encode the ingame location of the players. I found that this packet level information shows similar statistical properties that characterizes human motion models. My work focuses on the creation

of an algorithm which can automatically grab the dynamically changing fields of the protocols, especially the movement related payload segments of an unknown protocol. The requirements were to test player behavior in unknown protocols in an efficient way which resulted in a simplified character movement model comparing to player movement models in e.g., [41] for FPSs and [34] for MMORPGs.

Thesis 3. *[J2] I introduce a novel model and an algorithm for Deep Packet Inspection traffic classification based on dynamic byte signatures. [Chapter 4 of dissertation]*

4.3.1 Dynamic signature construction

The challenge was to specify the length of the fields, the direction of the fields (little-, big-endian) and to differentiate them from a simple increasing sequence. There can be use cases when other methods cannot deal with the content, e.g., packets with Protocol Header Encryption followed by a movement field. DPI methods dealing with static fields cannot recognize any fix signature in such packets.

Thesis 3.1. *[J2] I propose a new dynamic signature traffic identification method. The introduced algorithm exploits the spatial and temporal correlation by examining and extracting the correlation structure of the traffic and constructing signatures based on the observed correlation. The proposed algorithm considers the packets of the same flow as a time series and examines the H-parameter of the values of the different byte positions.*

The result of my model (see Thesis 3.2) was that I can consider the coordinate values as the result of a fractional Brownian motion (fBm) with Hurst (H) parameter close to 1. Therefore there is a strong correlation between the values of the same fields in the time series. The same test should be performed during the search for this structure in an unknown traffic.

The algorithm searches for fBm-process candidates. I examined how the test results altered if the original fBm process is decomposed to sum of byte position components ($fBm \sim \sum_{n=0}^{i}(2^8)^n X_n$, where i is the length of byte representation of the value, is decomposed to the X_n components). With simulations I proved that the H-value of the original process is always higher than the decomposed one. This means that even if the variance of the tested fBm process is so low that it falls in the range of a smaller component, considering all components the test results in a better fit for the original process meaning a higher H-value. These considerations are implemented in the proposed algorithm.

High H-parameter can be obtained by testing the timeseries with fix increments which are part of the gaming protocol message like sequence numbers or timestamps. The H-parameter testing has to be extended with the examination of the randomness

Application	Size	Signature	Endianess	Ratio
Age of Mythology [1]	10	?{1} CCCCCCC{7} SS{2}	B	19%
Command & Conquer 3 [2]	12	FFFFCCCCCCC{11} ?{1}	L	25%
Command & Conquer 3	16	???W{4} CCCWW{5} CC{2} ?CCC?{5}	L	18%

Table 1: A few example of the application signature patterns /C-strong correlation, F-flag, S-sequence number, W-walk, ?-not specified/

of the candidate byte positions to differentiate a fBm-process and a sequence number. To do this, the Relative Uncertainty (RU) (for details see Thesis 3.2) of the differences of the coordinates in a sliding window is calculated, and if the value is less than 0.8 then it is a sequence number with fix increments otherwise if RU is high it is considered as a moving packet. The sliding window is necessary as neighboring values can be fix in case of walking behavior – size of the step –, but the coordinate distance differences in a sliding window show randomness due to the movement in several dimensions. To differentiate the sequence numbers from the fix values in a specific position, the original time series is grouped into bins – the number of bins is selected in the function of the examined range of values – and the relative uncertainty is calculated for the binned values. If the RU is low then the examined values can be regarded as fix values.

Finally, the still empty fields are filled with the longest non-walking type fields from the state and H-parameter test tables.

Evaluation. The method was validated with a common-practice applied in other papers [25], [31] as well by analyzing open protocols, the RTP and the Gnutella protocol, which have several dynamic fields in the protocol. As a further proof-of-concept several proprietary gaming traffic and other known traffic types have been examined with the introduced algorithm and the existence of the fBm-like structure is proved to exist in several cases. A few example of the results can be seen in Table 1.

4.3.2 Player movement model

Real world human walk models, (e.g., [33]) can be applied to player behavior in gaming environment. However, the control environment (joystick, keyboard, mouse) may limit the player behavior. My goal was to construct such a model which a) can be efficiently tested, b) grab the characteristics of coordinate changes of human motion, c) use few model parameters and d) not game specific. Complex multidimensional models which describe human behavior even more accurately are not suitable for this task.

Thesis 3.2. *[J2] I propose a model for player movements based on two independent fractional Brownian motion (fBm) processes. The application of the model provides the input parameters for the dynamic signature traffic identification of Thesis 3.1.*

Based on the main dependence characteristics of the character movement process my presumption was that fractional Brownian motion (fBm) can describe the changes of character movements. The intuition behind the fBm model was my expectation that the character position coordinates exhibit high positive correlations. Therefore a player keeps going in the same direction with high probability, thus a simple random walk (e.g., Brownian motion) cannot describe its behavior.

I have chosen two independent fBm models, where one fBm would be fitted for the X and another would be fitted to the Y coordinates. The modeling of the Z coordinate was neglected due to the fact that the terrain serves as a constraint in this case.

Evaluation. The model provided parameters for the dynamic DPI method (see Thesis 3.1).

The filtered gaming traffic of an operational broadband network was analyzed with the R/S test [44] and multifractal analysis via wavelet-based methods [14]. First I applied an R/S test [44] for the increments. The test resulted in that the average estimated H parameter is 0.73 for both the X and Y coordinates calculated from 200 independent time series. This step was cross-validated by repeating the H-parameter estimation with the wavelet based method on the same time series. The estimated average H-parameter is 0.95 for X coordinate and 0.96 for the Y coordinate time series. An additional validation step has been performed by applying the wavelet based H-estimators on the original cumulant time series. The average H estimation resulted in 0.89 for the X coordinate and 0.9 for the Y coordinate time series.

Therefore my presumption that the character movements can be well modeled with two independent fractional Brownian motion for the X and Y coordinates has been validated by the statistical test results of real data. An appropriate H parameter is the average of all the estimation results from R/S, increment logscale and cumulant logscale analysis over 200 samples. It resulted in $H = 0.9$.

4.4 Wire speed traffic classification with parallel architecture

I discuss necessary components of a Graphical Processing Unit (GPU)-assisted traffic classification method, which is capable of multi-Gbps speed traffic classification on commodity hardware.

The wire-speed traffic classification with commodity hardware has several limiting factors starting even from the read out of the packets from the Network Interface

Card (NIC) ending up to the traffic classification of the packets. Current commodity hardwares have lot of resources – even usually unutilized for network related tasks like the GPU –, but it is difficult to exploit them with common techniques. To fully exploit the capabilities the exact knowledge of the architecture is necessary.

According to [16], the most resource consuming task is signature matching in traffic classification systems. String matching can be accelerated with ASICs, FPGAs [21] and previous generations of videocards [27, 28] (as texture operations), but they are difficult to modify and extend with new signatures and functions, which would be essential for traffic classification systems.

In [6], the deterministic finite automate (DFA) and the extended finite automate (XFA) based signature matching was analyzed. The authors found that the G80 GPU implementation was 9x faster than the Pentium4 CPU implementation. They emphasized the problem that data structure of the automate (in the order of MBytes) does not fit in the cache of current GPU architectures which would be essential for optimal operation. The packet processing speed of this solution was about 80,000 packets/sec (with FTP, SMTP and HTTP signatures).

In [42], the signature matching of an Intrusion Detection System (IDS) was done with GPU providing system throughput of 2.3 Gbps with synthetic traces and 600 Mbps with real traffic. In [43], Vasiliadis et al. further developed the work of [42] by reimplementing the DPI engine to the new CUDA architecture [3]. The overall system throughput increased by 30% on real traffic compared to [42]. [45] further refined the problem of load balancing the signature matching procedure of a large dictionary for multiprocessors. They proposed pattern set partition and input text partition together to fully utilize GPUs during the pattern matching.

Note that in [6, 42, 43, 45] the main tasks were to implement IDS on the GPU. This requires (a) the recognition of a huge set of protocol signatures with their numerous exploitation signatures, (b) find the signature anywhere in the byte stream, and (c) false positive hit is not allowed. I.e., the proposed methods for traffic classification task should not be directly compared to the above methods. As it can be seen later, my methods outperform them in raw packet processing speed, but on a more focused problem set of traffic classification.

Thesis 4. *[J3] I propose an efficient parallel processing solution for traffic classification. [Chapter 5 of dissertation]*

4.4.1 DPI with GPU

In order to achieve high processing speed with a GPU-based solution, the memory access overhead should be minimized. My goal was to store the signatures on a minimized memory footprint to achieve this. State machines are processor-efficient methods for signature matching but not space-efficient [6]. Use of hashes results in data

(a) Input application signature dictionary (S)

App#1	a?b
App#2	aaa
App#3	?a?
App#4	??a

(b) Bitmask of applications signatures (B)

101
111
010
001

(c) Alphabet-position dictionary (α)

	0	1	2
a	1100	1011	1000
b	1010	1110	1001

(d) Encoded signature database (E)

0101
1111
1011
1000

Table 2: Input data of the DPI method

compression, but it is difficult to perform wild-card character matching. E.g., typical solutions store the same signature with all the possible wildcard values in the hash [20]. This results in the boom of the hash table and in false positive hits as well.

The fast cached memory of the GPU is small, thus to make it possible to store a high number of signatures compression is desirable. The compression has to support wildcards.

Thesis 4.1. *[J3] I propose a memory optimized technique for signature matching. Due to the compactness of the hashing algorithm the data sets of the proposed DPI method can reside in the fast cached memory providing high efficiency on architectures supporting massively parallel computations.*

To encode the application signatures I propose a modification of the Zobrist hashing algorithm [48]. The dictionaries are stored in the cache of the GPU and apply the same encoding to check which application a packet may belong to. The proposed string matching technique uses the following input data (see Table 2 for illustration). The encoded signature database of a particular application is calculated by XORing together the codes of its characters if their corresponding bitmask is 1. In general it can be written as $E_i = \bigoplus_{j=0}^{|S_i|-1}(\alpha^j_{S^j_i} \bullet B^j_i)$. For notations and illustration, let us see a specific example based in Table 2: e.g., `a?b` is encoded into: $1100 \oplus 1001 = 0101$.

DPI data structures in GPU memory architecture. The *global memory* (uncached) space is filled with the array of network packets. During the initialization of each thread, the corresponding array of the packet bytes are copied from the global memory to the registers or to the *shared memory* (cached) of the thread. Thus global memory access does not reduce the speed of arithmetic calculations with the same data.

The *constant memory* space is cached so a read from constant memory costs one memory read from device memory only on a cache miss, otherwise it just costs one read from the constant cache. The pre-calculated input data structures are loaded into the constant memory space (the alphabet-position dictionary, the bitmasks and the encoded signatures). The allocable constant memory size is 64 Kbytes for the

	regexp	CPU	GPU	GPU_shared
Average [thousand packets]	14	72	138	1844

Table 3: DPI performance comparison – Packets processed per second

whole kernel in the test configuration. If we consider my example implementation where the signature database consists of 4 byte long values, then about 10 thousands of signatures could fit into the constant memory.

Evaluation. The method is capable of compressing a set of regular expressions and doing wire-speed DPI.

The compression gain is straightforward to estimate as fix sized strings are projected to fixed size strings. Table 3 summarizes the performance tests focusing on the average packet processing speed. Summarizing the results, the GPU-based shared memory version of the proposed method introduces a performance increase of two orders of magnitudes compared to the original regular expression based method. The average packet processing speed of the GPU is approx. 1,800,000 pkts/sec. This speed can be translated into network line speed by calculating with 500 byte average packet size which results in 6.7 Gbps speed.

4.4.2 Packet aggregation with GPU

Some specific traffic types are difficult to be identified simply based on DPI without considering flow groups and their history. E.g. according to [30], in case of small P2P flows trying to establish connection to non-existent peers, partially encrypted P2P traffic or Skype. In this section I propose how to combine efficiently the DPI and port based classification methods for flow classification on GPU.

Thesis 4.2. *[J3] I propose a high-speed aggregation and update method to support connection pattern-based identification based on parallel processing. The method aggregates all the information necessary for the classification of flows into one data structure providing the best-available hint all the time.*

The data structure of the proposed method is the Dedicated Port Table (DPT) (see Figure 10). The purpose of the DPT is to store the application hints of host-port pairs based on their communication history and provide the traffic classification engine with the best available hint all the time. The DPT stores the number of times a specific host-port pair was identified as the source or the destination of a specific application. The DPT is represented as a list of records containing the number of hits per application for each host-port pairs. In order to avoid a per-packet update

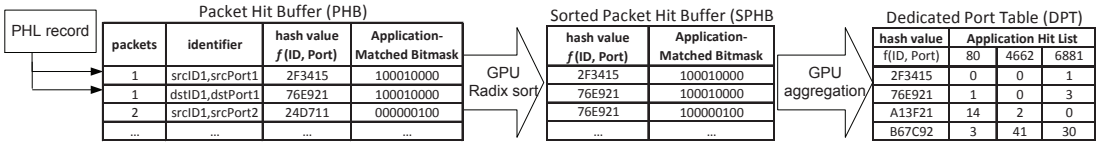


Figure 10: Dedicated port search

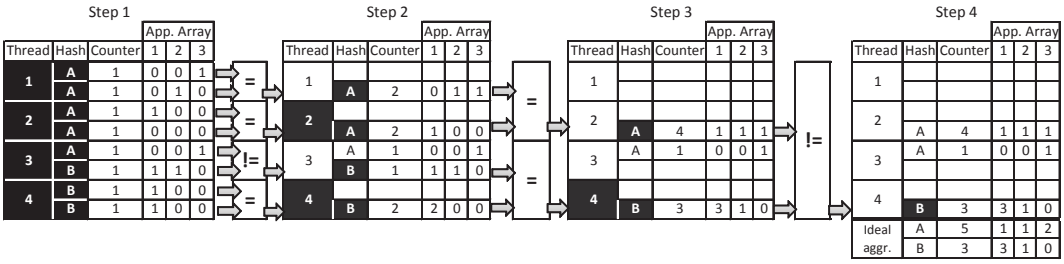


Figure 11: The aggregation of packet information with parallel reduction

of DPT, I propose to aggregate this information in the GPU by an extension to the parallel prefix scan [26]. This method provides massive parallelization and high efficiency with GPU (see Figure 11).

The input of the algorithm is the SPHB extended with a counter in each row. This counter is initialized to 1 and it represents the number of successfully aggregated packets (i.e., rows). In the first step in Figure 11, each thread compares the hash keys of the two rows they are responsible for. If the keys are equal then the aggregation occurs by summing the aggregation counters and the proper application hit array elements. The results are stored at every second row. In the following step, every second thread does actual work, comparing and adding data in the recently updated rows, and storing them in every fourth array position. This is repeated until there is no more row left untouched. The advantage of parallel reduction with such a data structure is that there are no concurrent memory reads or writes. The threads have to be synchronized after each step to ensure memory content consistency.

Evaluation. Beside the proposed use case, the method can be a building block of a wire-speed multiprocessor flow aggregation module further offloading the CPUs.

The compression ratio of the aggregation can be measured as the size of the Aggregated PHB divided by the size of the SPHB. In an example from a live traffic trace, the SPHB contained 256,000 rows. By ideal aggregation, APHB could be aggregated into 4,560 rows meaning that the compression ratio is 1.78%. By using the GPU, the APHB could be aggregated to 41,095 rows meaning a compression ratio of 16%. However, the aggregation ratio can be improved by repeating the

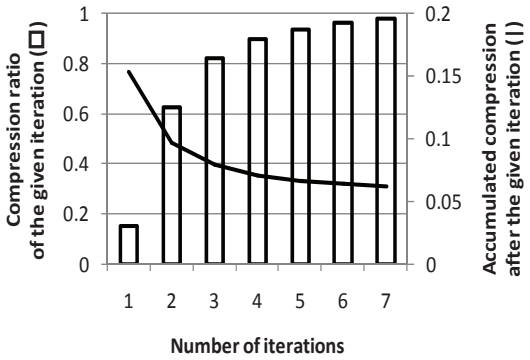


Figure 12: The compression ratio of the GPU based aggregation as a function of execution iteration number

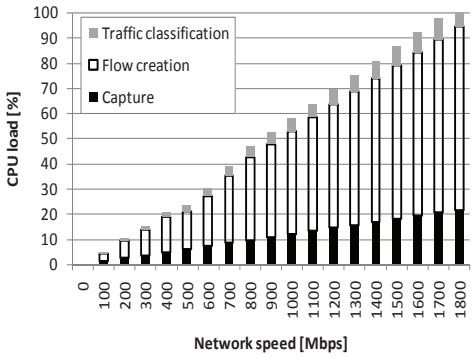


Figure 13: Total CPU load as a function of network speed

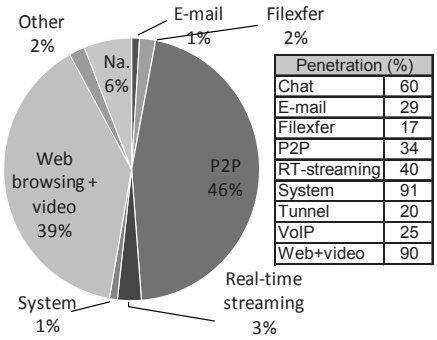
aggregation several times. The compression ratio as a function of the number of iterations is showed in Figure 12. It can be seen that the initial compression ratio can be approx. halved in the next few iterations resulting a total compression ratio of 8%.

The next question is the performance of the aggregation, i.e., how much time the aggregation needs. My experiments show that the aggregation tasks of the GPU are more than 4 magnitudes faster than the DPT update done by the CPU (hash insert and update). That is, the aggregation causes no practical overhead in the overall traffic classification process.

Considering the total system throughput with a multicore environment, the achievable speed scales up to the number of processors linearly achieving an approx. 6.7 Gbps total system throughput in a 4 core environment (see Figure 13).

5 Application of the Results

The presented methods can be used for analysis of traffic mixture of operational networks. A result of such analysis is given below. The analyzed trace is a three day long measurement collected in an operational mobile broadband network. Figure 14 shows the traffic mix in volume. The types of applications used by subscribers have a large influence on the network traffic. It can be seen that in the network the web browsing and P2P application take most of the bandwidth. Due to the fact that web browsing is downlink dominant, it contributes with fewer amounts to the uplink direction. The share of the P2P traffic and the web traffic stays constant during daytime and the share of the P2P traffic comparing to other traffics grows for the night and early morning hours. The hours of the growth of P2P traffic volume share



Penetration (%)	
Chat	60
E-mail	29
Filexfer	17
P2P	34
RT-streaming	40
System	91
Tunnel	20
VoIP	25
Web+video	90

Figure 14: Application volume share

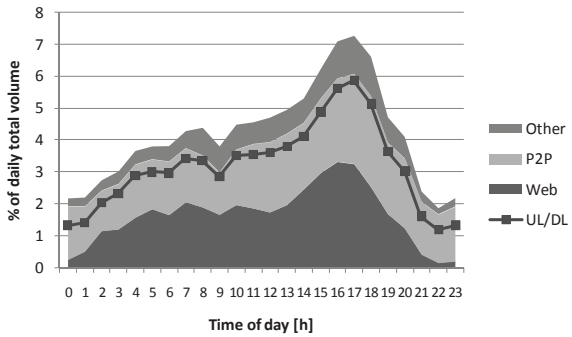


Figure 15: Upstream/downstream profile over time

coincide the hours of the growth of the uplink traffic volume share. Apart from the web and the P2P traffic, there is a considerable amount of streaming traffic. Due to the fact that the streaming traffic is downlink dominant, it only contributes to the downlink traffic share. Other applications such as e-mail or FTP give only a few percent of the total traffic.

Figure 15 shows the upload/download traffic ratio over time. The uplink/downlink volume share shows the ratio of data sent and received by subscribers. The share of the uplink traffic stays constant slightly above 1% of the total traffic during daytime and grows for the night hours. This is the effect of the missing of web application during night which is downlink dominant, thus its traffic does not contribute to the downlink traffic at night.

Besides the application volume share, it is also important to investigate what portion of the whole subscriber set uses a certain application regularly. In the right side of Figure 14 the application penetration can be seen. Browsing web is very common among the users, and this type of traffic generates system messages as visiting web pages generates DNS queries. Chat applications are very common among the users. It is interesting to note that only about 30% use e-mail applications, which is due to the popularity of instant messaging applications which are also capable to store the messages and send to the users when they become online, the other reason is the popularity of webmail services. In the consequence of these factors, the traditional e-mail protocols popularity has fallen back. From the figures we can notice that DNS (system) traffic is in correlation with web browsing, P2P traffic in does not generate significant DNS traffic.

Acknowledgements

The author is thankful for the invaluable help of his research supervisor Dr. Sándor Molnár PhD and all the colleagues from Ericsson TrafficLab and the Department of Telecommunications and Media Informatics.

References

[1] Age of Mythology. http://www.microsoft.com/games/ageofmythology/.

[2] Command and Conquer 3. http://www.commandandconquer.com/default.aspx.

[3] CUDA Programming Guide 2.0. http://developer.download.nvidia.com/compute/cuda/2.0-Beta2/docs/Programming_Guide_2.0beta2.pdf.

[4] IANA port numbers, http://www.iana.org/assignments/port-numbers.

[5] MSN Messenger. http://join.msn.com/messenger/overview2000.

[6] N. Goyal, J. Ormont, R. Smith, K. Sankaralingam, C. Estan: Signature Matching in Network Processing using SIMD/GPU architectures. http://www.cs.wisc.edu/techreports/2008/TR1628.pdf.

[7] RFC 2246. http://www.ietf.org/rfc/rfc2246.txt.

[8] RFC 4251. http://www.ietf.org/rfc/rfc4251.txt.

[9] Skype. http://www.skype.com.

[10] World of Warcraft. http://www.worldofwarcraft.com/index.xml.

[11] Matlab, 1983-2009. http://www.mathworks.com/.

[12] Lua Programming Language, 1993-2008. http://www.lua.org/.

[13] Silkroad Online, 2005. http://www.silkroadonline.net/.

[14] P. Abry and D. Veitch. Wavelet Analysis of Long-Range-Dependent Traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998.

[15] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic Classification On The Fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.

[16] J.B.D. Cabrera, J. Gosar, W. Lee, and R.K. Mehra. On the statistical distribution of processing times in network intrusion detection. In *In 43rd IEEE Conference on Decision and Control*, pages 75–80, Dec 2004.

[17] F. R. Cecin, J. V. Barbosa, and C. F. R. Geyer. A communication optimization for conservative interactive simulators. In *IEEE Communications Letters Vol. 10, No. 9*, pages 686–688, 2006.

[18] K. Chen, P. Huang, C. Huang, and C. Lei. Game Traffic Analysis: An MMORPG Perspective. In *NOSSDAV '05*, New York, USA, 2005.

[19] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic Classification through Simple Statistical Fingerprinting. *SIGCOMM Comput. Commun. Rev.*, 37(1):5–16, 2007.

[20] L. Deri. High-speed dynamic packet filtering. volume 15, pages 401–415, New York, NY, USA, 2007. Plenum Press.

[21] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Deep packet inspection using parallel bloom filters. *Hot Interconnects*, pages 44–51, Aug 2003.

[22] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. In *Proc. MineNet '06*, New York, NY, USA, 2006.

[23] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Perform. Eval.*, 64(9-12):1194–1213, 2007.

[24] S. Fernandes, R. Antonello, J. Moreira, and C. Kamienski. Traffic Analysis Beyond This World: the Case of Second Life. In *NOSSDAV '07*, Urbana, Illinois, USA, June 2007.

[25] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *MineNet '05*, New York, NY, USA, 2005.

[26] M. Harris, S. Sengupta, and John D. Owens. Parallel prefix sum (scan) with cuda. *Hubert Nguyen, editor, GPU Gems 3*, Aug 2007.

[27] N.-F. Huang, H.-W. Hung, S.-H. Lai, Y.-M. Chu, and W.-Y. Tsai. A gpu-based multiple-pattern matching algorithm for network intrusion detection systems. Okinawa, Japan, Mar 2008.

[28] N. Jacob and C Brodley. Offloading ids computation to the gpu. Washington, DC, USA, Dec 2006.

[29] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport Layer Identification of P2P Traffic. In *Proc. IMC*, Taormina, Sicily, Italy, October 2004.

[30] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proc. ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 2005.

[31] H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *SSYM'04*, Berkeley, CA, USA, 2004.

[32] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. In *Proc. ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 2005.

[33] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong. SLAW: A Mobility Model for Human Walks. In *Infocom'09*, March 2009.

[34] H. Liang, I. Tay, M. F. N., W. T. Ooi, and M. Motani. Avatar Mobility in Networked Virtual Environments: Measurements, Analysis, and Implications. *CoRR*, abs/0807.2328, 2008. informal publication.

[35] A. McGregor, M. Hall, P. Lorier, and A. Brunskill. Flow Clustering Using Machine Learning Techniques. In *Proc. PAM*, Antibes Juan-les-Pins, France, April 2004.

[36] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proc. SIGMETRICS*, Banff, Alberta, Canada, June 2005.

[37] M. Perényi and S. Molnár. Enhanced Skype Traffic Identification. In *Proc. Valuetools07*, Nantes, France, 2007.

[38] D. Pittman and C. GauthierDickey. A Measurement Study of Virtual Populations in Massively Multiplayer Online Games. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for Games*, pages 25–30, New York, NY, USA, 2007. ACM.

[39] S. Reid and M. Reid. The correlates of war data on war: An update to 1997. In *Conflict Management and Peace Science, 18/1*, pages 123–144, 2000.

[40] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. Second Annual ACM Internet Measurement Workshop*, November 2002.

[41] S. A. Tan, W. Lau, and A. Loh. Networked Game Mobility Model for First-Person-Shooter Games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for Games*, pages 1–9, New York, NY, USA, 2005. ACM.

[42] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, pages 116–134, Berlin, Heidelberg, 2008. Springer-Verlag.

[43] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis. Regular expression matching on graphics hardware for intrusion detection. In *RAID '09: Proceedings of the 12th international symposium on Recent Advances in Intrusion Detection*, pages 265–283, 2009.

[44] W. Willinger, M. Taqqu, and A. Erramilli. A Bibliographical Guide to Self-similar Traffic and Performance Modeling for Modern High-speed Network, 1996.

[45] C. Wu, J. Yin, Z. Cai, E. Zhu, and J. Chen. A hybrid parallel signature matching model for network security applications using simd gpu. In *APPT '09: Proceedings of the 8th International*

Symposium on Advanced Parallel Processing Technologies, pages 191–204, Berlin, Heidelberg, 2009. Springer-Verlag.

[46] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proc. ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 2005.

[47] M. Ye and L. Cheng. System-performance modeling for massively multiplayer online role-playing games. *IBM Syst. J.*, 45(1):45–58, 2006.

[48] A. L. Zobrist. A hashing method with applications for game playing. *Tech. Rep. 88, Computer Sciences Department, University of Wisconsin*, 1969.

Publications

Journal papers

[J1] A. Callado, **G. Szabó**, B. P. Gerö, J. Kelner, S. Fernandes, D. Sadok, : Survey on Internet Traffic Identification and Classification, *IEEE Communications Surveys and Tutorials*, 2009, Vol. 11, Num. 3, pp. 37–52.

[J2] **G. Szabó**, A. Veres, S. Molnár: On The Impacts of Human Interactions in MMORPG Traffic, *Journal of Multimedia Tools and Applications*, 2009, Vol. 45, Num. 1-3, pp. 133–161.

[J3] **G. Szabó**, I. Gódor, A. Veres, Sz. Malomsoky, S. Molnár: Traffic Classification over Gbit Speed with Commodity Hardware, *IEEE Journal of Communications Software and Systems*, 2010, Vol. 5, Num. 3.

[J4] **G. Szabó**, S. Molnár: Nagyon sok szereplős online szerepjátékok skálázási tulajdonságainak vizsgálata, *Híradástechnika*, 2007, Vol. LXII., Num. 11, pp. 20–26.

[J5] **G. Szabó**, S. Molnár: On The Scaling Characteristics of MMORPG Traffic, *Híradástechnika*, 2008, Vol. LXIII., Num. 1, pp. 40–47. (Selected Papers)

Conference papers

[C1] **G. Szabó**, I. Szabó, D. Orincsay: Accurate Traffic Classification. In *Proc., IEEE Wowmom 2007*, Jun 17-21, 2007, Helsinki, Finland.

[C2] **G. Szabó**, D. Orincsay, B. P. Gerö, Sándor Györi, Tamás Borsos: Traffic Analysis of Mobile Broadband Networks In *Proc., ICST Wicon 2007*, Oct 22-23, 2007, Austin, Texas, USA.

[C3] **G. Szabó**, D. Orincsay, Sz. Malomsoky, I. Szabó: On the Validation of Traffic Classification Algorithms In *Proc., PAM 2008*, April 29-30, 2008, Cleveland, Ohio, USA.

[C4] **G. Szabó**, A. Veres, S. Molnár: Effects of User Behavior on MMORPG Traffic In *Proc., ICC 2009*, June 14-18, 2009, Dresden, Germany.

[C5] **G. Szabó**, A. Veres, S. Molnár: Towards Understanding the Evolution of Wars in Virtual and Real Worlds In *Proc., ICSNC 2009*, September 20-25, 2009, Porto, Portugal.

Other non-related publications

[J6] **G. Szabó**, B Bencsáth: DHA támadás elleni védekezés központosított szűréssel, *Híradástechnika*, 2006, Vol. LIX. Num. 2, pp. 42–49.

[J7] A. Szentgyörgyi, **G. Szabó**, B. Bencsáth: Bevezetés a botnetek világába, *Híradástechnika*, 2008, Vol. LXIII. Num. 11, pp. 10–15. (Pollák-Virág díjas).