FDPW'99 Volume 2, 1999

TCPconan: A System with Flexible Management of TCP Connections Data Processing

Dmitri G. Korzoun[†], Dr. Iouri A. Bogoiavlenski[‡]

[†,‡]Department of Computer Science, University of Petrozavodsk
and
[‡]Institute for Informatics and Mathematical Modeling
of Technological Processes, Kola Science Centre, Apatity

Lenin St., 33, Petrozavodsk, Republic of Karelia, 185640, Russia

E-mail: {ybgv, dkorzun}@mainpgu.karelia.ru

Abstract

This paper presents a prototype of a system TCPconan—a TCP connections analyzer developed and implemented by the authors. The general issues of such systems design are discussed, as well. The key principles of the system are (i) TCP connections are treated as elementary units of traffic, and (ii) a specification language TCPconal provides the system with a mechanism of flexible data processing under user control.

1 Introduction

Traffic data processing is one of the necessary stages of distributed systems analysis. The problem of software development for this purpose is under intensive elaboration over the world. Many well-known companies take active steps in this direction. Let us mention only Cisco Corporation with its system NetFlow and a number of applications based on it, as a striking example [1].

The research was partially supported by grant N 96–42–36 of the Ministry of Secondary and Professional Education of the Russian Federation.

There are two essential issues that one should consider when designing such a software system:

(i) huge volumes of raw data;

(ii) a large number of various unpredefined characteristics and formats of their presentation that may be in need of a user depending on analysis purposes.

This paper presents the first prototype of a system TCPconan, which aims to help a researcher in solving a wide spectrum of problems based on processing TCP traffic data. Two key ideas make the base of the system: using connections as elementary units of traffic monitoring, and flexible management of data processing by a specification language.

The first one gives an effective way of processed data reduction. Moreover, this idea is in close agreement with the TCP paradigm—the most popular transport protocol widely spread throughout the current Internet.

The second idea is that a user should be allowed to control data processing according to his/her goals. For this purpose, the specification language TCPconal was designed, and it provides an effective mechanism of user management of traffic data processing. A TCPconal translator was built into the system and it is one of the vital parts of the system.

The system operates in the Linux operating system environment, and it is implemented in C. To build the prototype we used GNU C compiler, `flex`, `bison` and `tcpdump`. It is worth noting that the prototype implementation is fully based on free available software.

The rest of the paper is organized as follows. Section 2 contains a review of used methods and ideas. The architecture of the system TCP-conal is described in section 3. A short discussion of the first experiments is presented in section 4.

2 Software for TCP traffic monitoring and processing: the general features

2.1 MCSP class of software systems

Diverse problems that arise in the network management area demand development of a special class of software systems that perform the efficient fixation of network and application resources utilization. Any system from this class ought to implement the following four functions: traffic monitoring (capturing), data compression, storage and processing. Accordingly, we name such a system as a MCSP system and the corresponding class the MCSP class.

Development of a MCSP system is not a trivial problem. The potential market of the MCSP software grows together with the growth of volumes of traffic data requiring the analysis. This is a result of the growth of communication channels throughput. Leading providers of network hardware and software do active actions in this sector of the market. It is worth noting that they begin to intensively explore the idea of treating a connection as an elementary unit of traffic data.

The main functions of a MCSP system allow us to consider it as a certain complex of the following subsystems:

1. capture of traffic data;
2. data compression;
3. compressed data storage;
4. data processing;
5. report generation.

Methods of traffic capturing have been well developed by a lot of researchers. Any network operation system implements a set of appropriative functions, and modern network hardware support them.

Problems of traffic compression and storage become more and more topical, because volumes of transfered data grow and can reach extremely large values [2, 3, 4]. One approach, which is based on a notion of a connection as an elementary unit of traffic data, seems to be a perspective solution of these problems. The idea of connections monitoring instead of raw packets allows to compress traffic data in an order of several magnitudes without losing the most important information for the posterior analysis.

The problem of traffic data processing as well as report generation essentially depends on a research goal. The most of modern software (see for example [5–11]) do not support flexible management of traffic data processing. All calculated characteristics are fixed inside the software program and a user cannot explicitly vary them depending on the analysis he/she requires. Development of a special class of languages, intended to specify required processing and a format of the output results, is an effective way to implement the idea of flexible management that is a solution to many problems of this part of a MCSP system.

2.2 TCP traffic

TCP is a transport protocol, and it is the most popular one in the current Internet. It provides reliable data transfer. The TCP standard was described in [12], but some modifications was introduced in [13].

TCP traffic is a target of processing in the TCPconan system. This option was chosen for two basic reasons:

1. TCP traffic is the largest part of the current Internet traffic, so its behavior is the most important from a researcher point of view.
2. A notion of a connection was initially built into the TCP paradigm, and it allows to use naturally the idea of connections monitoring.

Thus, TCP is the main factor that determines a general character of the Internet traffic and is the best candidate for the base of an investigation, which aims to capture the key features of the Internet traffic.

A level of TCP connections is more important and adequate than a level of TCP packets, because it gives a clearer and more compressed picture of traffic behavior, rejecting many minor details and random events. Unfortunately, the level of TCP packets continues to be popular and it is used in the most part of modern software.

2.3 Definition of a connection

Generally speaking, a connection is defined as any substream of the total packets stream. To concretize this definition one should formulate the certain rules to extract (identify) the substreams.

There are some standard characteristics that may form the base of the connection identifying rules:

- Endpoints
- Establishment and termination
- Directionality
- Transfered data description
- Auxiliary attributes

2.3.1 Endpoints

Any connection has two members: A and B, each of them is called an endpoint of the connection. The most popular way to identify an endpoint is to consider a pair (a, p), where a is an IP address of a host and p is a port. This definition has the source in TCP and UDP nature [15, 16].

Another way is to use more aggregate strategy. For example, as an endpoint one can consider a group of hosts or a group of application protocols.

Thus, in accordance with research goals, a connection definition requires establishing a reasonable midpoint between granularity and aggregation of network objects. A high level of aggregation results in a high level of data volume reduction, but this declines the quality of saved information, because some important facts may be lost after such data compression.

The TCPconan system scans TCP connections, so it associates an endpoint with a pair of a TCP address and a port. However, some connections may be united into groups by filters defined in a TCPconal program.

2.3.2 Establishment and termination

The characteristics define the connection living time. One should determine when a connection is established and when it is terminated. A connection is said to be established when the first transmission[1] between the endpoints was observed. A definition of a connection termination is more complex. There are several ways to determine when a connection is terminated.

The protocol paradigm. Any connection–oriented protocol has rules to determine connection termination. For instance, TCP uses a notion of an abstract state machine, and each state corresponds to some current state of a TCP connection. However, this approach needs additional actions to perform the rules of a certain protocol. Also, it may become too complicated when the traffic is formed by different protocols on the same level. Moreover, there are popular protocols that are not connection–oriented, such as UDP. The main feature is in the possibility of capturing

[1]As a rule, it means the first packet.

some specific information about a connection. For instance, one can capture a status of TCP connection termination: normally finished, reset, aborted, etc. Thus, this method allows showing a more adequate picture of real traffic behavior.

Timeout on an inactivity period. In this method a certain value Δt is fixed. Let t_0 be the time of the last transmission between the endpoints. If there are no another transmission during the interval $[t_0, t_0 + \Delta t]$, then the connection is treated as terminated. The value Δt is called a timeout threshold. A problem of choosing an appropriate threshold Δt is not easy and can vary from some seconds to some minutes or even hours. As a rule, the threshold is a configurable parameter of the software. This method is easy to implement, so it is the most popular at the present time.

Limitation of the living time. While a connection has been established but not yet terminated, some data about this connection are accumulated in temporary memory. To reduce the volume of memory utilization, a method similar to the previous one is used. Again, the threshold Δt is fixed, but now it is a limit of the connection living time, a connection is not allowed to live more than time Δt. So a memory entry for a certain connection is taken only for the fixed time Δt, after that the connection is just treated as terminated, and the entry is freed.

In practice, these methods are used in combination. For example, TCPconan implements the first one, but there is a threshold (the default value is now equal to 3600 sec.) on an inactive period of any established connection. Cisco NetFlow combines the 2nd and the 3rd methods [1].

2.3.3 Directionality

Connections can be unidirectional or bidirectional. In the former case a connection transfers data only from one endpoint to another, and this fact is reflected as:

$$\text{either } A \to B\,, \text{ or } B \to A\,.$$

The latter one means that data may flow in both directions $A \rightleftarrows B$. The bidirectional case also has two variants: either data flows in the different directions are considered separately, or both streams are united and treated as indistinguishable.

Unidirectional connections are easier for monitoring and processing, because they need simpler data structures and shorter program code. It is the reason for the popularity of this technique in such hardware like routers [1].

The bidirectional method is more adequate to compute a real picture of traffic behavior, because the dependences between the orthogonal data flows are not lost. A good example is NeTraMet [17]—a free software implementation of the RTFM meter and manager [15].

The TCPconan system considers a connection as duplex, because it was stated by the TCP paradigm. Moreover, it distinctly processes data transmitted in the same direction: proper data, acknowledgments and retransmissions.

2.3.4 Transferred data description

The aim of a connection is to transfer data from one endpoint to another. Thus, there is a need to have a way of this data description.

Let $d(t)$ be some function that characterizes the data flow evolution during the living time of a connection. For each time point t the corresponding value $d(t)$ indicates how the connection data were transmitted in the period from the beginning of the connection till the moment t. This function may have vector values from $\mathbb{R}^n$, and n is the number of distinct parts of the data: one may separately consider traffic in different directions, or distinguish auxiliary and proper data. The simplest case is when a connection transfers homogeneous data, and it gives that $n = 1$.

A particular example of $d(t)$ is a function that shows a volume of data transferred to the moment t. It is a data volume accumulator. If the dynamics of data that flow through a connection is unknown, but the total volume V of transferred data is given, then we can introduce the function $d(t)$ as a linear accumulative function:

$$d(t) = \frac{t - t_1}{t_2 - t_1} V \ , \quad \text{when } t_1 \leqslant t \leqslant t_2 \ ,$$

where t_1 and t_2 are times of connection establishment and termination respectively.

The TCPconan system describes the connection data by a vector v of constant scalars, each of them characterizes the total amount of a certain type of transfered data. This goes according with data direction (from a client or from a server), measure units (in octets or in packets), and sort of data (proper data, acknowledgmented or retransmitted). This vector does not depend on time, but, as mentioned above, one can always construct the linear accumulative function $d(t)$.

2.3.5 Auxiliary attributes

A connection may have some auxiliary attributes, and it depends on a concrete target problem. In general, auxiliary attributes describe some peculiarities of the connection behavior.

For example, TCPconan stores the following attributes for any TCP connection:

- Establishment status: standard, no the first SYN, no the second SYN, no any SYN, simultaneous SYNs, not established.
- Termination status: finished by a client, finished by a server, aborted by a client, aborted by a server, timeout threshold exceeded, TCP-conan was interrupted.
- Service number (application protocol over TCP).

2.3.6 A general definition of a connection

Summarizing all the above statements, we conclude that any connection can be seen as a tuple $(P, T, d(t), a)$, where $P = (p_1, p_2)$—endpoints of the connection, $T = (t_1, t_2)$—its start and finish times, $d(t)$—a function that characterizes the data stream, transferred through the connection, and a—some auxiliary attributes.

The directionality property is taken into account by function $d(t)$. In some cases auxiliary attributes a can also be included in $d(t)$ (it will only increment a dimension of the vector space, where $d(t)$ takes its values).

2.4 Specification of traffic data processing

A type of traffic analysis strongly depends on research goals and cannot be fixed once and forever. As a result, different algorithms of data processing are needed to satisfy various research requirements. Therefore, a MCSP system has to provide user management of data processing and results reporting.

The simplest way to implement this mechanism is a command line, when a user controls the system with a certain set of options. However, as practice shows, the modern features of data processing are so rich and diverse that such a poor tool as a command line has not been enough to express in an easy form the most part of user requirements.

Another approach is based on a language of data processing specification. It means that a user writes a program, where the required processing and a report format are defined. The specification language approach has the most expressive power of currently existing ones. One example of a specification language is SRL [14], and its compiler is available as part of NeTraMet [17].

The system TCPconan supports a new specification language—TCPconal, which is a tool to specify the required processing of TCP connections data and a desired format of the reports. Some basics of the TCPconal description was introduced in [18, 19].

The authors believe that the specification language approach is one of the most perspective ways to implement the idea of flexible management of traffic data processing.

3 Architecture of the TCPconan system

We named our system TCPconan, and it is an abbreviation from "**TCP con**nection **an**alyzer". This system is destined to investigate TCP traffic on the level of TCP connections, and its first prototype is oriented to capture the following characteristics:

- Summary of general information on TCP connections, which were established by a given group of hosts.
- Frequency characteristics of activity for each member of a given group of hosts relative to the total TCP traffic.
- Frequency characteristics of utilization for each application protocol under TCP.

3.1 Principles and a skeleton scheme of the system

The system pursues two goals: the first one is to capture and to store data on TCP connections, the second is to calculate and to output some characteristics of TCP traffic. These goals divide the system in two parts: data acquisition and data processing.

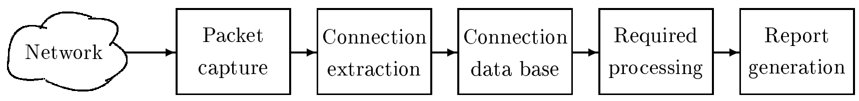

Figure 1. *Scheme of the system TCPconan*

TCPconan is a set of utilities; each of them can be used either independently, or in combination with others by pipes. The system uses TCP connection as an elementary traffic unit. TCPconan is oriented to deal with some structural formations, which may appear in a target network: various combinations of hosts, clients, servers and application protocols. The TCPconal translator is supported by the system to control traffic data processing and formats of reports.

Accordingly with the definition of a MCSP system, TCPconan is partitioned into five main subsystems, as is shown in Figure 1. Each subsystem is implemented with one or two modules. Each module can be piped with another software if its input data format is correctly kept.

The prototype of TCPconan consists of 6 modules. A module Capture is used for data acquisition (TCP packet capture). The TCP connection extraction belongs to the duties of modules Machine and Pconan. The connection database was implemented as a module Condb and it keeps connections data in a compressed form. The crucial part of processing and reports generation are performed by a module Conan. A user interface is a function of the module Interface.

3.1.1 Packets capture

The module Capture captures all TCP packets, which flow through the monitored channel and it keeps a small amount of information per packet for subsequent processing. This information includes:

1. Time stamp. This is the time of the packet observation by Capture (year, month, day, hour, minute, second, fraction of a second).
2. Endpoints of TCP connection (IPv4 addresses and TCP port numbers of a client and a server).
3. Volume of transferred data in octets.
4. Sequential numbers of transferred and ack'ed data octets.

5. TCP flags (SYN, FIN, RST, PUSH, URG, ACK).
6. TCP slide window size.
7. TCP options.

All these data are written in a file (default extension `.raw`). At present the module Capture uses the standard Unix utility `tcpdump` [20], which outputs the data in a different format than the TCPconan one. Translation from tcpdump format to TCPconan one was implemented with a utility `flex`. The goal of TCPconan format is effective memory saving, because the tcpdump format has a lot of minor details.

The module Capture saves captured packets in two formats: textual and binary. The textual format uses only decimal digits, spaces and EOLs. It provides the ability to compress effectively. The binary format uses a computer–oriented form of data presentation, and it is more effective with respect of the memory utilization.

The module checks errors in the input data format. Each record of a packet with the wrong format is ignored and the warning is written to a log file.

Figure 2 introduces an example of external channel monitoring and shows the role of the module Capture in this process.

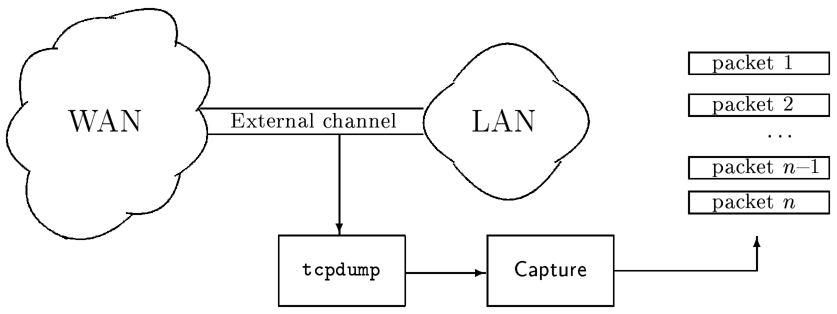

Figure 2. *An example of the TCPconan approach of an external channel monitoring*

3.1.2 TCP connections discovering

The module Machine emulates a TCP state machine, which is a formal description of a TCP connection. Input data have the same format as the output of Capture.

The module scans the input data (TCP packets stream) and renews a special table, which contains information on the discovered TCP connections. For each live connection, the table contains an entry, which is accumulated information on this connection including the current state of the corresponding TCP state machine.

An entry in the table are allocated immediately after the first packet of a connection has been observed by the module Machine. The module does not remove any entries—this is a task of the module Pconan, which shares the table with Machine.

Memory resources are limited, so the table has entries for a finite number of TCP connections. The maximum number of TCP connections, which are concurrently processing at the same time, may be configured.

A log file warns of all observed errors.

3.1.3 Preliminary processing of TCP connections

The functions of the module Pconan are the following: (i) determination of terminated TCP connections, (ii) their preliminary processing, and (iii) form and output the report.

The module looks through the table, filled by the module Machine, trying to search a terminated connection. When it succeeds, it processes the data accumulated for this connection by Machine, and then it writes the result into a file. The file, generated by Pconan, contains the following data for each connection:

1. Addresses of connection participants
2. Time stamps of connection start and finish
3. Application protocol (type of service)
4. Connection status (establishment and termination character)
5. Volume of transferred data (data sent and received, retransmitted and actually transferred, in octets and in packets)

The entry occupied by the processed connection is freed.

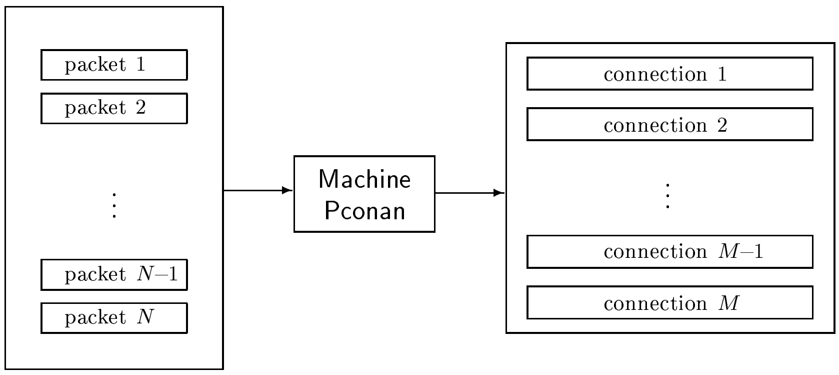

Figure 3. *Scheme of the joint work of modules* Machine *and* Pconan

From a functional point of view, the modules Machine and Pconan are an analogy of the `tcpreduce` [5] and `tcptrace` [6] utilities. For the same reasons as Capture, the module Pconan supports two output format: binary and textual.

Machine and Pconan may work only together and the scheme is shown in Figure 3.

3.1.4 Connections data base

The module Condb implements a special database for storing data on TCP connections. The database allows the following services:

- Searching connections with a given filter
- Deleting old data
- Adding new data

Filters are described in a TCPconal program, and the rules of their constructing are based on the specification of this language, see [18] or [19] for details.

New data are formed by the module Pconan and they are sent to Condb by a pipe. The general scheme of data extraction from a database with the module Condb is shown in Figure 4.

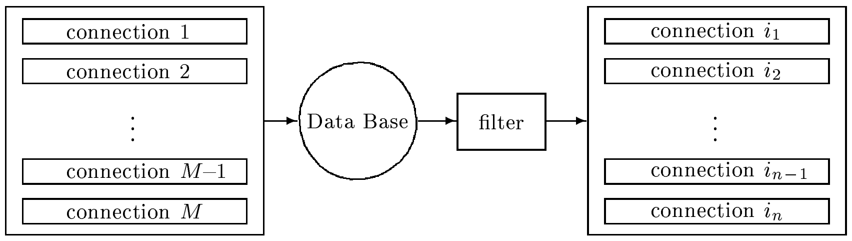

Figure 4. *Connections data extraction from a data base using a filter.*

In the first prototype of the system the database is a set of flat files: for each day of monitoring there is a directory, which contains the corresponding raw files generated by Pconan. The utility `gzip` is used to reduce a volume of data.

3.1.5 Basic processing of TCP connections

The module Conan is aimed to perform more complex and resource consuming processing of the connections data, stored in the database, and they are available through Condb.

In the first prototype the following types of processing are implemented:

- General summary on TCP connections of a given group. This includes a description of each connection (client, server, application protocols, living time, data volumes, etc.).
- Frequency analysis of given group activity. This allows, for instance, to discover the most active members of the group.
- Frequency analysis of application protocol utilization by a given group. This allows, for instance, to discover the most popular application protocol used by the group.

The module Conan implements a TCPconal translator. Thus, the module is managed by a TCPconal program: a user writes a TCPconal program as a text file and it is an input parameter of the module Conan. Figure 5 shows a scheme of a user interaction with Conan.

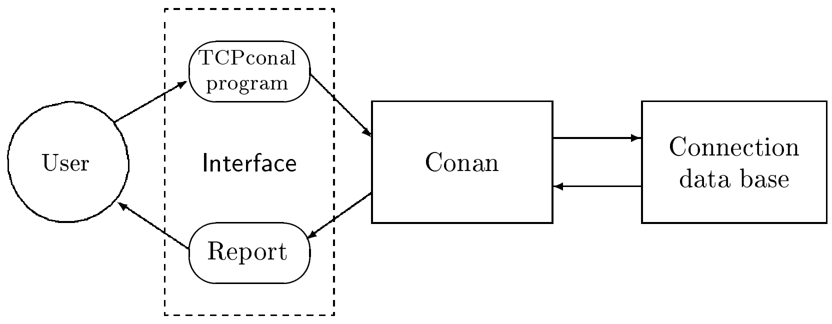

Figure 5. *Scheme of the modules* Conan *and* Interface

3.1.6 The user interface

The module Interface allows a user to interact with the whole system. A user gives a request to Interface. The module sends the request to Conan and awaits the results. The results are given back to the user as a report on executed processing. All these steps are pictured in Figure 5.

In the first prototype, a report is a text file. The format of a report is described in a corresponding TCPconal program.

3.2 Modules interactions

The modules interact by data exchange. This means using pipes: the output data of one module are the input data for another one.

There are two module combinations, recommended at this time in the first prototype:

- Filling a connections database from a stream of TCP packets (data acquisition part of TCPconan)
- Data processing in accordance with users' requirements (data processing part of TCPconan)

The first of them has the following scheme:

`tcpdump 'tcp'` $\rightarrow$ Capture $\rightarrow$ Machine&Pconan $\rightarrow$ Condb

The symbol '&' means here that both the modules Machine and Pconan work simultaneously. So, `tcpdump` captures TCP packets from a network. The module Capture translates them into the TCPconan format. The modules Machine and Pconan discover and process TCP connections. Then the module Condb stores them into the data base.

The second combination has a shorter scheme:

$$\text{Condb} \rightarrow \text{Conan} \leftrightarrows \text{Interface}$$

The module Condb takes connections data and sends them to Conan. Conan, in accordance with a TCPconal program from Interface, performs the required processing and then gives the results to Interface. The last one forms a report for the user.

4 Experiments

We have performed a number of experiments to test the system TCP-conan and TCPconal language. One Ethernet segment was chosen as a testbed. This segment contains hosts belonging to the mathematics and physics departments of the University of Petrozavodsk. The prototype was installed on a host `delta.cs.karelia.ru` (AMD K5 processor at 166 MHz).

The whole TCP traffic in the range of some hours was investigated. The following types of analysis were performed:

- Connections analysis
- Hosts Activity Analysis
- Application Protocols Usage Analysis
- Intruders monitoring

As the main result, we had that the most popular applications protocols in the target Ethernet segment are WWW (40%) and ftp-data (15%). The most active host is `proxy.karelia.ru` which is a proxy server for WWW and ftp traffic. Among these we discovered that there was an aggressive behavior of host `k143.karelia.ru` from the Department of Physics. This host scanned some TCP ports of host `epsilon.cs.karelia.ru` from the Department of Computer Science.

A more detailed description of all these experiments can be found in [18, 19].

5 Conclusion

The prototype of a new MCSP system TCPconal was introduced in this paper. A purpose of the system is to perform various types of TCP traffic data processing flexibly defined with specification language TCPconal. These problems often arise in distributed systems investigation. Today, TCP traffic analysis becomes one of the most popular approaches for network management.

The implemented TCPconal translator provides flexibility and expandability for setting data processing and specifying a report format. These features are sure to be very useful for any kind of research in this area.

The implemented prototype is oriented to solve three types of problems: connections analysis, frequency analysis of hosts activity, and frequency analysis of application protocols usage. All of them are everyday tasks in network management and their goal is the estimation of the general state of the network and its resources. Also, they may be considered as an initial base to solve various types of problems, which may arise in a network research.

References

[1] *NetFlow Services and Applications.* Cisco White Paper. 1998.

[2] T. Monk, and K. Claffy, *Internet Data Acquisition and Analysis: Status and Next Steps.* National Laboratory for Applied Network Research, USA. 1997.
`http://ftp.uni-mannheim.de/ftp/info/inet_97/f1/f1_3.htm`

[3] L. Press, *Tracking the Global Diffusion of the Internet.* CACM, November 1997. Vol. 50, 11, pp. 11–17.

[4] V. A. Ponomarev, I. A. Bogoiavlenski, and T. Alanko, *An Ethernet Segment Perfomance and Workload Characterization Using Set of Filters Based on Free Software Tools.* Proceedings of FDPW'97-98, Vol. 1. University of Petrozavodsk, 1998. pp. 130–152.

[5] V. Paxon, *TCP-reduce documentation.*
`http://ita.ee.lbl.gov/html/contrib/tcp-reduce-doc.html`

[6] S. Osterman, *TCPtrace Home Page.*
`http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html`

[7] V. Paxon, *Automated Packet Trace Analysis of TCP Implementations.* Proceedings of SIGCOMM'97, Sept. 1997.

[8] M. Mathis, and J. Mahdavi, *Diagnosing Internet Congestion with a Transport Layer Performance Tool.* 1996.

[9] Y. Murayama, and S. Yamaguchi, *DBS: a powerfool tool for TCP performance evaluations.* 1997.

[10] *Netperf: A Network Performance Benchmark.* Hewlett-Packard Company, 1995.

[11] P. B. Danzig, and S. Jamin, *tcplib: A Library of TCP Internetwork Traffic Characteristics.* Report CS-SYS-91-01, Computer Science Department, University of Southern California, 1991.

[12] J. Postel, *Transmission Control Protocol.* RFC 793. 1981.

[13] R. Braden, *Requirements for Internet Hosts—Communication Layers.* RFC 1122, 1989.

[14] N. Brownlee, *SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups.* RFC 2723, 1999.

[15] N. Brownlee, C. Mills, and G. Ruth, *Traffic Flow Measurement: Architecture.* RFC 2063, 1997.

[16] K. Claffy, *Internet Traffic Characterization.* PhD Thesis, University of California, 1994.

[17] N. Brownlee, *Traffic Flow Measurement: Experiments with NeTraMet.* RFC 2163, 1997.

[18] D. G. Korzoun, and I. A. Bogoiavlenski, *TCPconal: A Prototype of Specification Language for TCP Connections Data Processing.* Proceedings of FDPW'97-98, vol. 1, University of Petrozavodsk, 1998. pp. 212–237.

[19] D. G. Korzoun, and I. A. Bogoiavlenski, *Flexible management of TCP connections data processing.* Transactions of University of Petrozavodsk on Applied Mathematics and Computer Science, vol. 8. University of Petrozavodsk, 1999. To appear. (in Russian)

[20] V. Jacobson, C. Leres, and S. McCanne, *Tcpdump documentation.* `http://noc.ucsc.edu/cie/Topics/56.htm`