Formalization of the quality of service mechanisms in packet-switched networks

Andrey Khobnya, Viktar Liauchuk, Oleg Demidenko

***Abstract*—**This paper presents analysis and generalization of quality of service (QoS) mechanisms in packet-based networks. Decomposition of QoS mechanisms is presented. Each QoS mechanism can be split into four components each of which is responsible for the defined subtask. Method of formalization is presented in which each QoS component is presented by function of the defined arguments. This method was used for simulation of QoS mechanisms in next generation networks.

***Keywords*—**network traffic, QoS, simulation.

I. Introduction

The objectives of the simulation of the quality of service mechanisms in networks can be the following:

– optimization of the quality of service (QoS) control mechanisms settings in order to achieve better parameters of network performance;

– calculation of the quality of service characteristics using different settings in order to determine the reserves to increase the network load;

– determination of influence degree of various traffic classes to the network;

– calculation of the threshold load at which the quality of service will be provided under the current settings;

– research of existing and development of new quality of service mechanisms.

At present the formalization of the quality of service mechanisms in packet-switched networks as a reasonably accurate analytical models is an impossible task. An alternative approach is the simulation of complex systems. Simulation is a research method in which the investigated system is replaced by a model with sufficient accuracy to describe the real system which conducted experiments aiming to the study of this system. This paper presents the analysis, generalization, formalization and simulation method for quality of service mechanisms in data networks with packet switching. The presented method of generalization can be used to build specific simulation models as well as a platform for simulation and research of quality of service mechanisms.

II. Overview of the quality of service mechanisms

Let's consider the quality of service mechanisms used in networking equipment. One of the largest manufacturers of communications equipment split provided QoS tools to *Congestion Management* tools and *Congestion Avoidance* tools. Congestion Management tools include FIFO Queueing, PQ, CQ, WFQ, CBWFQ and LLQ. Congestion Avoidance tools include WRED, DWRED, Flow-Based WRED and DiffServ Compliant WRED. FIFO Queueing provides basic store and forward packets. This is the simplest algorithm for processing the queue, which in some cases can be used by default and does not require configuration. PQ allows one to set strict priorities for critical traffic. PQ guarantees the primary processing of the

A. Khobnya, Francisk Skorina Gomel State University, Gomel, Belarus (e-mail: andrey@khobnya.me).
V. Liauchuk, Francisk Skorina Gomel State University, Gomel, Belarus (e-mail: liauchuk@gmail.com).
O. Demidenko, Francisk Skorina Gomel State University, Gomel, Belarus (e-mail: demidenko@gsu.by).

most important traffic. Priorities for packages can be flexibly defined depending on ports, source and destination addresses, packet size, etc. CQ reserves a predetermined fraction of the total bandwidth of the network interface for each given type of traffic. If any type of traffic does not use dedicated amount of bandwidth than the other types of traffic use it instead. WFQ splits traffic into multiple streams based on parameters such as source address, destination address, source and destination port, etc. Each traffic stream is of applied priority (or weight). CBWFQ extends the functionality of WFQ providing the ability to configure traffic classes. CBWFQ allows one to select the exact percentage of bandwidth for each class. Up to 64 different classes of traffic can be configured. LLQ allows one to give the absolute priority to one traffic class and adjust the proportion of bandwidth for other classes in CBWFQ. WRED is an implementation of the well-known algorithm RED which uses the IP Precedence to ensure the privilege of traffic with higher priority. DWRED is an extension of WRED and it allows one to set the minimum and maximum thresholds for the packets queue length of different classes of traffic. Flow-Based WRED and DiffServ Compliant WRED is also an extension of the algorithm WRED and it allows management to separate the destruction of packages for different flows and different traffic classes, respectively [1].

Other manufacturers of network hardware and software use similar concepts to build quality of service mechanisms. Also they uses the concept of HTB to drop and ensure the priorities of packets [2]. In general the quality of service mechanism in any network classifies incoming packets, distributes them across multiple stores or deletes some of them, then determines the sequence of sending packets from different stores.

III. Analysis and generalization of the quality of service mechanisms in packet-switched networks

After analyzing algorithms of quality of service mechanisms we can distinguish four main components of each QoS mechanism:

– packets queues;
– packet classification algorithm;
– active queue management algorithms;
– scheduling algorithm.

Packet queue is the simplest queue with sequential processing. The quality of service mechanism can use one or more packet queues. Generally the differences between the various mechanisms are in the other three components.

Packet classification algorithms provide the packet class determination and distribution of packets of different classes across different queues. Let’s consider the existing methods for traffic classification used in the quality of service mechanisms.

Classification by IP Precedence (ToS) is a classification method that uses a three-bit code from IP Precedence field of Type of Service (ToS) header in IPv4-package. This method allows one to split eight types of traffic based on priority.

Classification by DSCP is a classification method that uses a 6-bit code Differentiated Services (DSCP) in 8-bit Differentiated Services Field (Field DS) of IP-packet header. Theoretically the network may have up to 64 different classes of traffic with different values of the DSCP. It gives operators of telecommunications services greater flexibility in the definition of traffic classes.

Flow-based classification is a way of traffic classification which can use TOS, DSCP, IP-address of the sender, IP-address of the recipient, the sender port and destination port. It is used to distinguish different traffic flows in the network interface.

Classification by MPLS QoS is a traffic classification method to be applied to in cases when MPLS protocol is used. It uses a three-bit field Traffic Class which allows one to separate the eight traffic classes. Generally this method of classification is used in conjunction with the others. It is often used in next generation networks.

Deep Packet Inspection (DPI) is a classification method based on the analysis of package contents which allows one to separate traffic of different application level protocols and applications.

Summarizing the above mentioned facts, let's identify the following main features of traffic classification algorithms:

– usage of special marker fields of IP-packet header in order to identify the class of traffic (ToS, DSCP, MPLS QoS);

– usage of sender IP-address, receiver IP-address, sender port and receiver port;

– usage of information about the application level protocols and applications.

These properties need to be considered in constructing simulation models of the quality of service mechanisms.

The next important component of the quality of service mechanism algorithms is active queue management (AQM). Active queue management (AQM) is the class of algorithms that control the destruction of packet in network interface in case of an overflow or near overflow state of device internal buffer. It is necessary for reducing a load to a network. Let's consider the existing algorithms for active queue management used in the quality of service mechanisms.

Tail Drop is a simple algorithm for active queue management used to control the overflow. Unlike more complex algorithms Tail Drop does not differentiate the traffic and handles all of the packets in an identical way. When the size of the buffer queues packets reaches the maximum value, Tail Drop starts to discard all newly arriving packets. Admission of new packages will resume when the buffer has enough space for receiving incoming packets.

Random Early Detection (RED) is a more complex algorithm of active queue management designed to prevent network congestion. RED is focused on addressing the major drawbacks of the algorithm Tail Drop: uneven buffer allocation for different traffic flows and global synchronization of TCP-connections which leads to uneven pulsating network load. RED takes into account the average value of the queue size and drops packets based on statistical probabilities. If the buffer is almost empty then it takes all arriving packets. As the size of the queue increases and the probability of dropping, the incoming packet is raised as well. When the buffer is filled, the probability reaches one and, therefore, all arriving packets are dropped.

RED algorithm allocates resources more effectively than Tail Drop because of its being less sensitive to the negative effects of bursty traffic which itself occupies a small fraction of the bandwidth and does not allow it to block the activity of the other TCP-connections. More packets are sent from the source, then it is more likely that a particular packet from a given source will be discarded, since probability is proportional to the number of events in the data queue. Thus RED can be a problem solving of global synchronization of TCP-connections.

Weighted random early detection (WRED) is an extension of algorithm RED which enables one to set several different queue thresholds depending on the type of traffic. Thus the probability for packets is different for various classes. For example, the interface can have a low threshold for low priority traffic. When filling in the queue until this threshold, all packets with lower priority will be discarded. Thus WRED protects a high priority traffic in the same queue.

Adaptive random early detection (ARED) is an extension of algorithm RED which allows one to take the original algorithm more or less "aggressive" to the packets depending on the value of the average queue length. If the average queue length packet oscillates around the minimum threshold, the risk of dropping the packet increases. Conversely, if the average queue length is around the maximum threshold, the algorithm works more conservatively, i.e. probability of destroying the package is reduced.

Robust random early detection (RRED) is an extension algorithm RED designed to protect against network attacks such LDoS (Low-rate Denial-of-Service is denial of service due to its low speed). The basic idea is to identify and filter packet attacks before classical algorithm RED or its modifications will be applied to packets. LDoS attack packets are identified and discarded by this algorithm. As a rule, the sender TCP-packet starts to pause before sending a new packet if it detects a loss. Therefore, if the source sends a packet within a short period of time after the loss of a package sent to them, this sender can be the source of the attack. This idea is the basic one for the identification algorithm packages LDoS attacks inside RRED [3].

RED with Preferential Dropping (RED-PD) is an extension algorithm RED which uses the story of the destruction of broadband packages in order to identify flows.

Random Exponential Marking (REM) is an active queue management algorithm similar to RED, but it differs from its certain measures load and function of the probability of destroying packets. The dependence of the probability function of the load in REM is close to logarithmic. The function measures of the load depends on the difference between the total rate of incoming traffic and upstream bandwidth, as well as on the difference between the current queue length and a predetermined threshold. REM algorithm aims to provide high load and low losses.

Blue is an active queue management algorithm which differs from the RED algorithm that requires no setup by the network administrator. It is able to automatically learn and adapt. Blue maintains the value of the probability p and discards any incoming packet with a given probability. Whenever the queue overflows, p is increased by a small constant pd, and whenever the queue is empty, p decreases by pi < pd. If the composition of the traffic does not change, then the p-value slowly converges to the optimum and makes full use of the communication channel.

Stochastic fair Blue (SFB) is an extension algorithm Blue which allows one to separate different traffic flows and to maintain a variety of probability values for each stream. SFB provides uniform distribution of the buffer capacity for each flow of traffic.

Resilient Stochastic Fair Blue (RSFB) is an extension of algorithm SFB which is designed to protect against attacks, such as spoofing DDoS. This algorithm improves on the algorithm with SFB flow separation.

Sontrolled delay (CoDel) is an algorithm for active queue management designed to overcome the effect of excessive buffering (buffer bloat) on network interfaces by setting limits on the delay of arrival of the package. The algorithm works within certain intervals. Initial interval lasts for 100 ms. Time delay is calculated for each queued packet which is equal to the timeout of packet in the queue. Minimum time delay is saved. When the last packet is deleted from the queue interval providing that the lowest delay for the current time interval is more than 5 ms, this single packet is discarded, and the interval used for the next group of packets is reduced. If the lowest delay for the current interval is less than 5 msec, the packet is sent and the duration of the interval is reset to 100 ms. The interval is reduced in accordance with the inverse square root of the number of consecutive slots where the packages have been destroyed due to excessive delay values.

Active Virtual Queue (AVQ) is an active queue management algorithm based on the use of the virtual queue the capacity of which changes dynamically based on the rate of arrival of new packets and the smoothing parameter. It happens because the destruction of the packages should be more aggressive when the load exceeds the required interface and less aggressive when loading is below the required [4].

PI Controller is an active queue management algorithm based on the algorithm of proportional-integrator controller.

Summarizing the information discussed above, let's identify that active queue management algorithms are characterized by the following features:

– the ability of customization (most algorithms have customizable settings);

– management of only one queue;

– usage of the data about queue length and buffer capacity;

– usage of data about source, destination and other fields of the packet header (e.g., WRED identifies different traffic classes, SFB supports multiple traffic flows, etc.);

– storing and maintaining the internal state (e.g., Blue stores and maintains probability parameter, RED-PD stores history of packets, etc.);

– great number of algorithms involve the destruction of the package before putting it in place. However, for example, CoDel destroys packets when they are removed from the queue.

These properties need to be considered in constructing simulation models of the quality of service mechanisms. Also note that if the quality of service mechanism uses several queues for packets, then different algorithms of active management can be used for different queues.

Another important component of QoS mechanisms is the scheduling algorithm. The algorithms of this class control the sequence of packets to be sent. Let's consider the existing scheduling algorithms used to send packets designed to ensure the quality of service mechanisms.

First In First Out (FIFO) is the simplest scheduling algorithm for sequentially processing a single storage working in the principle of "first come - first out". When using this algorithm, traffic prioritization is not performed, i.e. all packets have the same permissions.

Priority Queuing (PQ) is the scheduling algorithm which works with absolute priority queues. It provides an absolute priority of a certain class of packets over other packet classes. As a rule, the implementation of this algorithm assumes the use of four queues:

– high;

– medium;

– normal;

– low.

Processing of packages from different queues performs sequentially. Prior to complete processing of all packets from a queue with a higher priority processing of a low-priority packet queue is not performed. In the presence of the incoming packet intensive real-time high-priority traffic the outer channel is monopolized by packages of this type of traffic, i.e. a high priority queue in this case will never be exhausted and all other traffic will be blocked. This is a major drawback of this scheduling algorithm.

Round-robin is the scheduling algorithm works by iterating by a circular loop. Queues are processed sequentially in each iteration and packet from each queue is sent.

Weighted round-robin (WRR) is an extension of round-robin scheduling algorithm which allows one to set different weights for different queues. WRR per iteration sends the queued packets directly proportional to the weight and inversely proportional to the average size of the last packet in the queue.

Weighted Fair Queuing (WFQ) is a scheduling algorithm which provides per-flow bandwidth share depending on its weight. This algorithm is a generalization of the algorithm fair scheduling (Fair Queuing - FQ). If a channel at a rate R is used for N streams, than the processing speed of each of them is R/N if Fair Queuing is used.

Unlike FQ, Weighted Fair Queuing allows to set up the fraction for each stream. Virtual starting time of $S(k, i)$ processing and virtual time of the end of $F(k, i)$ processing is assigned to each incoming packet $p(k, i)$, where k is the ordinal number of the packet, and i is the number of queue. Starting and finishing time is calculated as follows:

$$S(k, i) = max(F(k-1, i), V(a(k, i))),$$
$$F(k, i) = S(k, i) + L(k, i) / r(i), \quad (1)$$
$$F(0, i) = 0,$$

where $a(k, i)$ is the packet arrival time, $L(k, i)$ is the size of the packet, $V(t)$ is the virtual time function which is defined as multiplication of argument and reverted sum of active flows weights in current moment. The algorithm chooses the packet with the smallest virtual time of the end of the processing and sends it.

Low Latency Queuing (LLQ) is an queuing discipline with low latency. LLQ is an extension of WFQ algorithm with additional priority queue. Additional queue in LLQ enables handling of the most delay-sensitive traffic.

Summarizing the facts discussed above, let’s identify that the scheduling algorithms are characterized by the following features:

– the ability of customization (most algorithms have customizable settings);

– usage of data about state of all queues in QoS mechanism;

– storinge and maintaining internal state (e.g. WFQ stores and maintain parameters of the virtual time of arrival and sending packets).

IV. Formalization and building simulation models of quality of service mechanisms

From the point of view of the theory of queuing networks, the quality of service mechanism can be represented by a component to accumulate *transacts* before *service device* which is the outgoing interface of the network node. Packages are represented as service requests (or *transacts*). Each transact must contain the information necessary to simulate the operation of the quality of service mechanism and for the calculation of QoS characteristics: a virtual model time of packet sending, source port, destination port, IP-addresses of sender and recipient, the packet size, protocol type used by application layer, fields ToS, DSCP, MPLS QoS, and other information.

For the simulation of QoS mechanism it is necessary to describe its reaction on the occurrence of two discrete events: the new incoming packet arrives and the next outgoing packet can be sent.

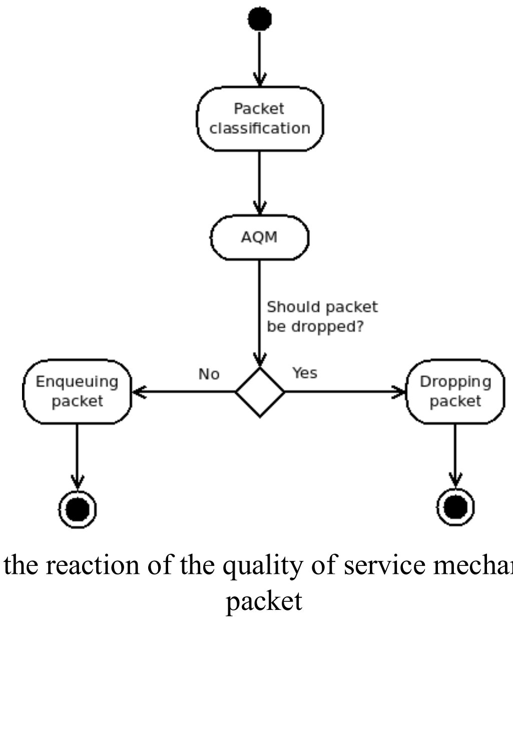

Fig. 1 Activity diagram of the reaction of the quality of service mechanism on the arrival of a new packet

Using the analysis above the reaction of the quality of service mechanism on the arrival of a new packet time can be represented as a UML-activity diagram indicated in Fig. 1. First, packet classification is performed to determine the queue to which it can be placed. Then mechanism of active queue management (AMQ) is applied to the selected queue to determine whether the packet must be dropped. The packet is then queued or dropped.

Using the above analysis, the reaction of the quality of service mechanism, when an event of the next outgoing packet can be sent, can be represented as a UML-activity diagram indicated in Fig. 2. The first step is performed by scheduling algorithm to determine the queue from which the next packet must be sent. Then the second part of the algorithm of active queue management is applied to determine whether the packet must be destroyed or not. The packet is removed from the queue. Then the packet is sent or deleted.

Algorithm of packet classification can be defined by function $f_c(P, S_c)$, where P is a tuple of parameters of packet, and S_c is the current state of classification component memory. This function can have values from set of all possible pairs (n, S'_c), where n is an ordinal number of queue which packet must be placed to, and S'_c is a new state of classification component memory. Algorithms of active queue management can be defined by functions $f_a(P, q, S)$, where P is a tuple of parameters of packet, q is a tuple of parameters of queue state, and S is the current state of active queue management component. This functions can have values from set of all possible pairs (r, S'), where r is the result of algorithm work which can be one of the boolean values, a S' is a new state of active queue management component memory. Scheduling algorithm can be defined by function $f_s(Q, S_s)$, where Q is a tuple of queues state values, and S_s is the current state of scheduling component memory. This function can have values from set of all possible pairs (n, S'_s), where n is the ordinal number of queue which packet must be sent from, and S'_s is a new state of scheduling component memory state. Thus the quality of service mechanism can be defined by parameters $n, f_c(P, S_c), f_{ai1}(P, q_1, S_1), ..., f_{ain}(P, q_n, S_n), f_{ao1}(P, q_1, S_1), ..., f_{aon}(P, q_n, S_n), f_s(Q, S_s)$, where n is a number of queues, $f_c(P, S_c)$ is a function of classification algorithm, $f_{ai1}(P, q_1, S_1), ..., f_{ain}(P, q_n, S_n)$ are functions of AMQ algorithms for each queue from 1 to n that will be applied to incoming packets, $f_{ao1}(P, q_1, S_1), ..., f_{aon}(P, q_n, S_n)$ are functions of AMQ algorithms for each queue from 1 to n that will be applied to outgoing packets, and $f_s(Q, S_s)$ is scheduling algorithm function.

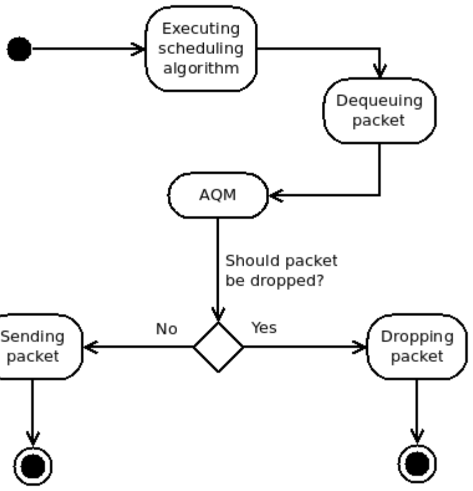

Fig. 2 Activity diagram of the reaction of quality of service mechanism on the next outgoing packet which can be sent

Using the concepts of object-oriented programming, the functions described above can be represented by the classes that implement the corresponding interfaces which contain a single method. This approach is used for building the problem-oriented tool for simulation automation of next generation networks. Functions of classification algorithms are represented by a class that implements the interface Classification which contains a single method classify(Packet packet) which returns the number of the queue to the packet to be placed. The function of active queue management algorithms is represented by a class which implements an interface ActiveQueueManagement containing a single method shouldBeDroped(Packet packet, Queue queue) and which returns a boolean value. The functions of scheduling algorithms are represented by a class which implements an interface Scheduling containing a single method schedule(List<Queue> queue) and which returns the ordinal number of the queue which packet must be sent from. The toolkit contains the implementation for modeling many existing algorithms of quality of service mechanisms and allows the user to define its own custom algorithms using programming languages Java, JavaScript, Jython, JRuby, Scala, Groovy, Clojure and some others.

For example, the implementation of a classification algorithm that uses only field MPLS QoS can be defined using Clojure language as follows:

```
(gen-class
   :name "MplsQosClassification"
   :implements [Classification]
   :constructors {[] []}
   :prefix "mpls-qos-")

(defn mpls-qos-classify [this packet]
  (if (< (.getMplsTrafficClass packet) 2)
    (.getMplsTrafficClass packet)
    (if (< (.getMplsTrafficClass packet) 4)
      2
      (if (< (.getMplsTrafficClass packet) 6)
        3
        4))))
```

Implementation of the modification of AVQ active queue management algorithm for packet arrival can be defined using Jython language as an example as follows:

```
class AvqArrivalManagement (ActiveQueueManagement):
    def __init__(self, alpha, gamma, queue):
        self.alpha = alpha
        self.gamma = gamma
        self.virtual_capacity = queue.getCapacity()
        self.busy_virtual_capacity = 0
        self.last_time = -1

    def shouldBeDroped(self, packet, queue):
        rate = packet.getSize() / (packet.getArrivalTime() - self.last_time)
        self.virtual_capacity = self.alpha
 * (gamma*queue.getCapacity() - rate)
        self.last_time = packet.getArrivalTime()
        if self.busy_virtual_capacity + packet.getSize() <
self.virtual_capacity:
            self.busy_virtual_capacity += packet.getSize()
            return False
        else:
            return True
```

The simulation system calls user-defined algorithms in sequence represented by activity diagrams in Fig. 1 and 2 upon the occurrence of the relevant event. The system simulates traffic generation of different types for the simulation of packet arrival events [5, 6].

This approach allows to describe the quality of service algorithms using high-level languages abstracting the details of the implementation of the simulation system. Decomposition mechanisms for QoS on four types of components allow one to create simulation models of new mechanisms by combining existing components in the system and/or redefining some of them.

V. Conclusion

This paper presents the analysis, generalization, formalization and simulation method for the quality of service mechanisms in data networks with packet switching. The proposed method is used to develop tool for automated simulation of next generation networks which allows to simulate and explore the quality of service mechanisms.

References

[1] Cisco IOS Quality of Service Solutions Configuration Guide. http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfintro.html. Accessed August 2015.

[2] Traffic Control HOWTO. http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/components.html. Accessed August 2015.

[3] C. Zhang, J. Yin, Z. Cai, W. Chen, "RRED: Robust RED algorithm to counter low-rate denial-of-service attacks", *IEEE Communications Letters*, № 14 (5), 2010, pp. 489-491.

[4] S. S. Kunniyur, R. Srikant, "An adaptive virtual queue algorithm for active queue management", *Networking, IEEE/ACM Transactions on*, №2, vol. 12, 2004, pp. 286-299.

[5] V. Kulinchenko, O. Demidenko, P. Chechat, "An approach to identify channel capacity of LAN on TCP/IP, ICMP and UDP protocols", *Problems of physics, mathematics and technics*, 2014, № 4 (21), pp. 1-3.

[6] O. Demidenko, A. Khobnya, "Conceptual model for the generation of VoIP traffic in the NGN", *Francisk Scorina Gomel State University Proceedings*, 2014, №6(87), pp. 117-122.

Co-funded by the Tempus Programme of the European Union

This publication is the result of the project implementation:
TEMPUS CERES: Centers of Excellence for young RESearchers.
Reg.no.544137-TEMPUS-1-2013-1-SK-TEMPUS-JPHES

CERES