METHOD OF SEARCHING FOR ROUTES WITH DELAY AND JITTER LIMITATION

Nadezhda Matveeva
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russian Federation

Keywords: *network, route, jitter, delay*

Abstract

Question of optimal routes creation for data transmission is very important in modern network systems. The quality of data transmission is affected by such parameters as delay and jitter. Delay is the time required to deliver a unit of data (usually a packet or message) through the network, measured as the elapsed time between the injection of the first bit at the source to the ejection of the last bit at the destination [1]. The jitter is the difference between the maximum and the minimum duration of an action (processing action, communication action)[2]. In this paper new algorithm of routes creation is proposed for systems with limits on delay and jitter values.

There are many different algorithms providing ways to find one shortest route or shortest routes between all graph nodes. Deikstra, Bellman - Ford and Lee algorithms [3] are well known and can be used to build only one shortest path between pair of nodes. But it is often necessary to transmit data with determined time parameters.

The proposed algorithm allows to generate data routes taking into account speed of data links. The effect of several data flows influencing each other during transmission over the same link is also evaluated. For such flows delay and jitter values can be estimated. For each generated route maximum delay and jitter values along the route are estimated. For each source - destination pair a route that conforms to the constraints defined by user is selected. If the network can't route data meeting user defined requirements, user receives information about failure.

As a result information about possibility of data transmission with required parameters can be obtained at the network design stage.

This algorithm allows to reduce network design time and evaluate upper limit of timing parameters before network hardware implementation. The same algorithm generates data routes in accordance with the requirements to transmit data in information system.

1 Introduction

Modern communication systems for data transmission are actively developing. Data rate increases. Requirements of data transmission are rising. In modern data networks, great attention is paid to algorithms and methods, which ensure and guarantee the data transmission characteristics. It increases performance and quality of service (QoS). Delay, jitter, throughput are critical parameters for modern computing systems. Problem of control jitter and delay value is considered in this paper.

Many techniques to ensure the requirements of jitter and delay values are already developed. That methods such as Virtual Clock, Jitter Virtual Clock, Core-Jitter Virtual Clock, Delay Earliest-Due-Date (EDD), Jitter-EDD, Stop-and-Go and etc.

Virtual Clock is a traffic control algorithm for high-speed packet switching networks [4]. The basic idea of Virtual Clock was inspired by Time Division Multiplexing (TDM) systems [5]. Jitter Virtual Clock is non-work-conserving version of the Virtual Clock algorithm. It uses a combination of delay-jitter rate controller and Virtual Clock scheduler [6]. Core-Jitter-Virtual

Clock (CJVC) is a variant of Jitter Virtual Clock, which does not require per flow state at core nodes. The key idea is to have the ingress node to encode scheduling parameters in each packet's header [6]. In classic Earliest-Due-Date (EDD) scheduling each packet is assigned a deadline and the packets are sent in order on increasing deadlines. Delay-EDD service discipline is an extension where the server negotiates a service contract with each channel. The contract states that if a source obeys a peak and average sending rate, then the server will provide a delay bound. The key lies in the assignment of deadlines to packet. The server sets a packet's deadline to the time at which it should be sent, had it been received according to the contract.[7] The Jitter-EDD discipline extends Delay-EDD to provide delay-jitter bounds. After a packet has been served at each server, it is stamped with difference between its deadline and actual finishing time. A regulator at the entrance of the next switch holds the packet for this period before it is made eligible to be scheduled.[7] The Stop-and-Go service discipline aims to preserve the 'smoothness' property of traffic as it traverses the network. Time is divided into frames. In each frame time, only packets that arrived at a server in the previous frame time are sent. [8] But these mechanisms do not allow to guarantee the boundary of these transmission parameters simultaneously.

2 Description of the proposed algorithm of searching for routes with delay and jitter limitations

2.1 Terms and Definitions

Link is physical channel between the two devices over which data is transmitted.

Data path/route is a sequence of nodes through which data flows are transmitted.

Data flow – data packets are transmitted between the specified source and receiver on a given path. The path does not change during the operation of the network.

Adjacency matrix – a way of representing a graph графа $G = (V, E)$ as a matrix $A_G = [a_{ij}]$ with size $|V| \times |V|$, where $a_{ij} = 1$ or $a_{ij} = w$. W – weight of the edge if $[v_i, v_j] \in E$ and $a_{ij} = 0$ otherwise. V – vertices of graph G, E – edges of graph G.

The proposed algorithm is designed for data transmission systems in which applications generate a data flow with a defined intensity. Priorities of the data flows are not supported. Information collection systems from the various types of sensors may be an example of such data transmission systems. For such systems is very important that jitter does not exceed some value. This value is determined by the system designer and depends on the requirements of the data processing. Select the type of address is not considered in this paper. Different addressing (logical addressing, regional logical addressing, path addressing and etc) are possible. The basic requirement is that paths should not be changed.

2.2 Theoretical calculations

There are some definitions of jitter in books. In this paper we will use the definition proposed by the Hermann Kopetz. Jitter (J) is the difference between the maximum and minimum data packet delay [9].

$$J_\mathrm{i} = Dmax_i - Dmin_i$$

i – data flow identifier, $Dmax_i$ – maximum packet delay of data flow i, a $Dmin_i$ – minimum packet delay of data flow i, J_i – jitter value of data flow i.

The minimum delay value is achieved in the case when the packet is passed to the output port of the switch without any delay waiting in a queue. Namely, packets from other data flows are not compete. Therefore the minimum delay consists of packet processing delay at the source and destination, switches and links.

$$Dmin_i = Ds_i + \sum_{l \in L_i} Dl_l + \sum_{s \in S_i} DminSw_s + Dd_i$$

Where l – link identifier, i – data flow identifier, s – device (terminal, switch) identifier, Ds_i – packet processing delay at the source, L_i – set of links that carry data flow i, Dl_l – delay transmission time on the link l, Dd_i

– packet processing delay at the destination, $DminSw$ – minimum packet processing delay at the switch.

Packet processing delay at the source depends on packet size, recording rate of one byte in the input buffer. Packet processing delay at the destination depends on packet size, recording rate of one byte in the output buffer. Recording rate of one byte in the buffer is defined hardware implementation.

$$Ds_i = Sp_i \cdot Tw_k$$

$$Dd_i = Sp_i \cdot Tr_k$$

k – source identifier of data flow i, Sp_i – packet size of data flow i, Tw_k – recording time per byte in the k device output buffer, Tr_k – recording time per byte in the k device input buffer.

Delay transmission time on the link depends on packet size, data rate.

$$Dl_l = Sp_i \cdot v_l$$

v_l – data rate of link l, Sp_i – packet size of data flow i.

The minimum packet processing delay at the switch is obtained when the desired switch output port available for data transmission.

$$DminSw_s = Sp_i \cdot Tr_s + Trt_s + Tp_s + Tarb_s + Sp_i \cdot Tw_s$$

Tp – time of transmission parameters calculation for incoming packet время расчета параметров передачи пришедшего пакета, $Tarb$ – arbitration operation time in the switch, Trt – header processing time.

Maximum packet delay occurs when packet in each switch waits until all competing packets from other flows will be passed through output port of the switch.

$$Dmax_i = Ds_i + \sum_{l \in L_i} Dl_l + \sum_{s \in S_i} DmaxSw_s + Dd_i$$

$DmaxSw$ – maximum packet processing delay at the switch.

$$DmaxSw_s = Sp_i \cdot Tw_s + Trt_s + Tp_s + Tarb_s + Sp_i \cdot Tw_s + \sum_{j \in F_s} (Sp_j \cdot Tw_s + Tarb_s)$$

F – set of competing flows to the output port of the switch in relation data flow i.

Applying basic mathematical operations we have concluded that jitter depend on size of packet competing flows, length of the data path, recording time per byte in the switch device output buffer and arbitration operation time in the switch. This formula allows us to estimate the upper bound of jitter.

$$J_i = Dmax_i - Dmin_i = \sum_{s \in S_i} DmaxSw_s - \sum_{s \in S_i} DminSw_s = \sum_{s \in S_i} \sum_{j \in F_s} (Sp_j \cdot Tw_s + Tarb_s)$$

Jitter depends on the logic of packet processing in communications devices, the arbitration algorithm and other hardware and software features of the system component implementation. On the data paths search, providing a predetermined limit on the jitter value in data transmission with guaranteed bandwidth is not affected. When using the mechanism for another class of systems is only necessary to calculate a formula for the upper bound of jitter.

2.3 Basic steps

Search algorithm paths the data according to the specified requirements is proposed in this paper. It allows evaluating the possibility of providing the limitations specified by the jitter amount in data transmission with guaranteed bandwidth. Transmitting path search algorithm based on the representation of the system in the graph. Terminal nodes and switches are represented as vertices, link between nodes – as edges. Representation of the system as a graph allows to use graph algorithms to find the shortest routes between two nodes, breadth-first search, depth-first search and etc [10].

Algorithm of searching data paths consists of several steps. It is represented in Fig. 1. The data system for which the paths are generated is represented in graph. A data transmission path for satisfying the bandwidth requirements is formed for each data flow. Upper bound jitter values are calculated for each flow in accordance with the obtained data path. Further calculated values are compared with the requirements presented for each data flows. If the upper bounds are less than jitter user requirements, the built paths provide jitter requirements for data transmission with guaranteed bandwidth. When no paths satisfying the bandwidth and jitter requirements, then data transmissions with user requirements are impossible in the system.

Modification and combination of algorithms on graphs [11] are used to find the data paths. After searching data paths set of data paths for all flows generated or information that it is impossible to build paths satisfying the user requirements is displayed to system designer.

2.4 Searching routes

A system description should be presented before the search routes. A system is considered correct if nodes, links, data flows, requirements are described. Next, a tree describing the existing routes satisfying the bandwidth requirements is constructed. Jitter calculation is carried out in case of a successful construction of at least one decision tree branch for all flows. Calculation based on formulas described above. The obtained values are analyzed for compliance with the specified user requirements. If the requirements are satisfied then algorithm finish. If all possible data paths were found and neither of them does not satisfy user requirements, then it is considered that the system specified requirements are not supported. This is schematically represented in Fig. 2.

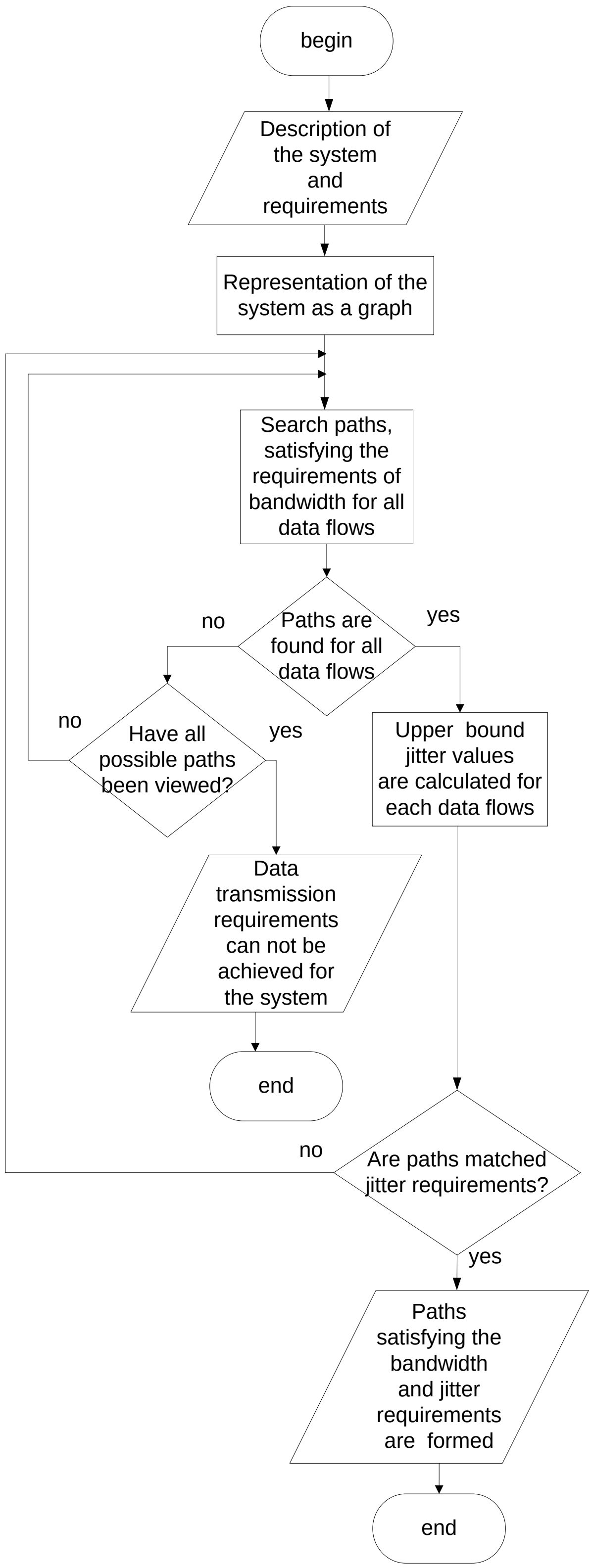

Fig. 1 Algorithm basic steps

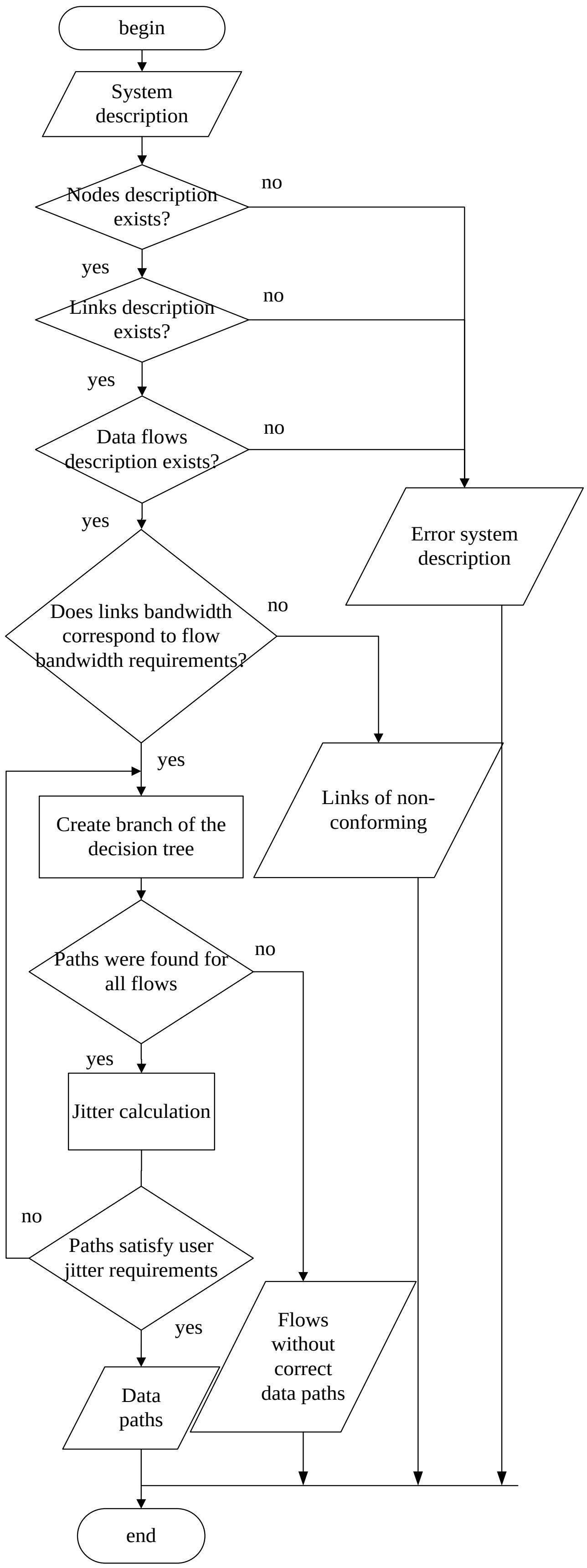

Fig. 2 Scheme generation mechanism data routes

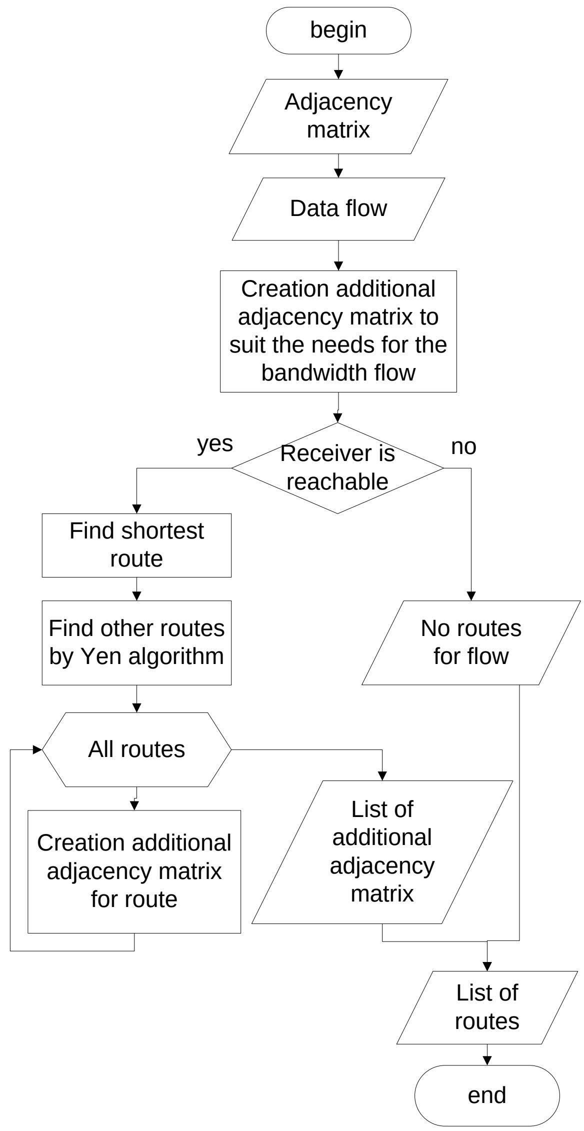

Fig. 3 Scheme of create decision tree branch

3 Evaluation

In this chapter we present several examples of systems, theoretical estimates of the upper bound of the jitter and simulation results.

3.1 Example 1

The first system is very simple. It consists of one source and one destination. Network structure is presented in Fig. 4. This network structure has been selected to show that in this case each flow has only one possible path for data packets transmission. In such a situation is obvious fact that two flows compete with each other. It causes jitter.

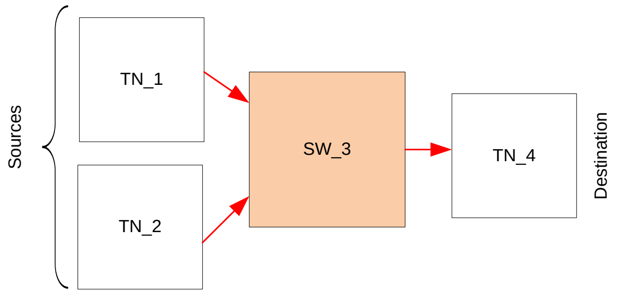

Fig. 4 Network structure 1

Two data flows with identical characteristics are transmitted in the network. Packet size is 32 bytes. Data rate on the link is 800 Mbit/s. Recording time per byte in a buffer is 10 ns. Arbitration operation time in a switch is 50 ns. Jitter requirement is 1000 ns.

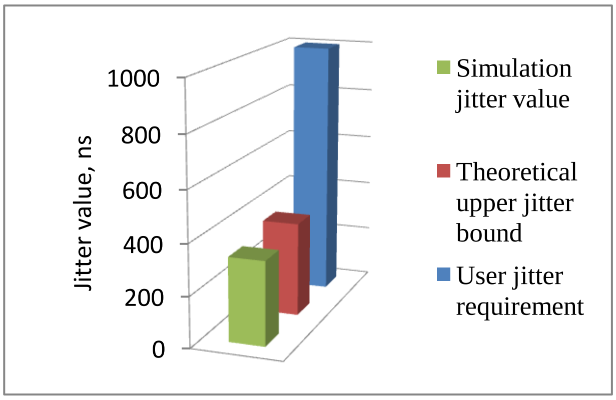

Fig. 5 Comparison of jitter for example 1

Applying formulas presented above we can obtain an upper jitter bound in the network. The upper jitter bounds are equal 370 ns for described data flows. Also network with presented configuration parameters was simulated using DCNSimulator[12,13]. Sources generate traffic with uniform distribution. The minimum packet processing delay is 1100 ns, the maximum – 1340 ns, jitter – 330 ns. User requirement jitter, calculated upper jitter bound and simulation value are shown in Fig. 5.

3.2 Example 2

The second system is more complex. It consists of 5 terminal nodes and 8 switches. Network structure is presented in Fig. 6.

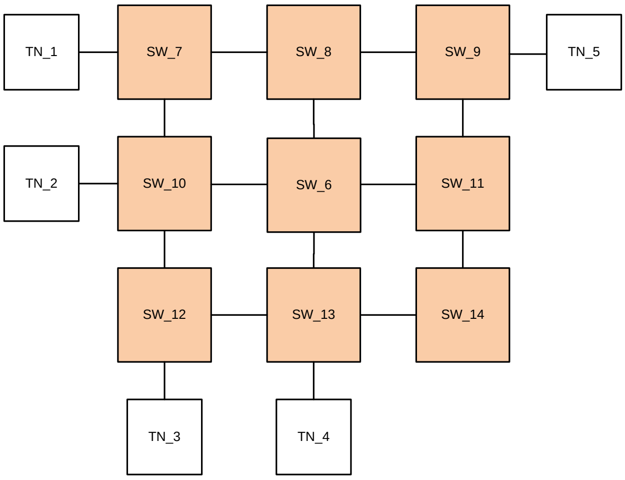

Fig. 6 Network structure 2

Five data flows with identical characteristics are transmitted in the network. Sources are TN_1, TN_2, TN_3, TN_4. Destination is TN_5. Packet size is 32 bytes. Data rate on the link is 800 Mbit/s. Recording time per byte in a buffer is 10 ns. Arbitration operation time in a switch is 50 ns. User jitter requirements for flows are different. Values are present in Fig. 7.

Flow identifier	User jitter requirement, ns
1	2000
2	2500
3	2500
4	3000

Fig. 7 User jitter requirements

Data transmission paths, which satisfy the user requirements, have been found by the proposed algorithm. Also upper jitter bounds were found for all flows. Results are presented in Fig. 8.

Flow identifier	Upper jitter bound, ns
1	1850
2	2200
3	2200
4	1100

Fig. 8 Calculated upper jitter bound

Generated paths are shown on Fig. 9.

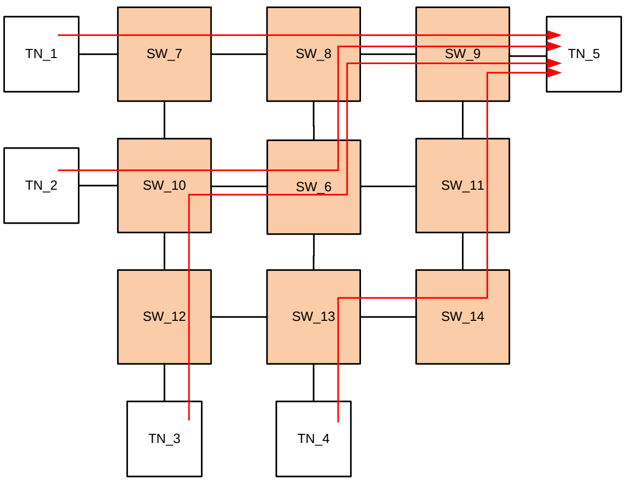

Fig. 9 Network structure 2 with correct data transmission paths

Similarly network with presented configuration parameters was simulated using DCNSimulator. Sources generate traffic with uniform distribution synchronously every 25 us. Simulation results are presented in Fig. 10.

Flow identifier	Minimum delay, ns	Maximum delay, ns	Jitter, ns
1	2330	2330	0
2	2990	2990	0
3	3650	3650	0
4	3320	3320	0

Fig. 10 Network structure 2 simulation results

User requirement jitter, calculated upper jitter bound and simulation value are shown in Fig. 11.

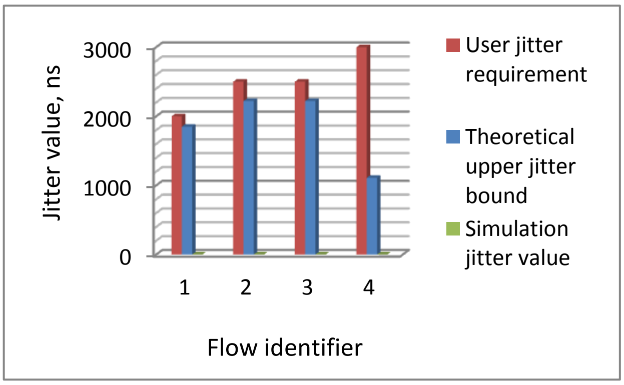

Fig. 11 Comparison of jitter for example 2

4 Conclusion

The algorithm of generating data transmission paths providing user jitter requirements with guaranteed bandwidth is presented in this paper. It allows evaluating the possibility of providing the limitations specified by the jitter amount in data transmission with guaranteed bandwidth. Basic algorithm principles are discussed in detail. Theoretical evaluation the time characteristics are presented with the formulas for calculating it. Also this material gives some network examples for generation data routes in accordance with customer requirements. Moreover networks were simulated using DCNSimulator and results were analyzed. Theoretical and simulation jitter values were comparisons with each other. It is shown that the simulation jitter value does not exceed the upper theoretically calculated bound.

This approach allows reducing network design time and evaluate upper limit of timing parameters before network hardware implementation. Furthermore algorithm generates data paths in accordance with user requirements to transmit data flows in system.

Acknowledgment

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under grant agreement n° 13.G36.31.0003.

References

[1] William James Dally, Brian Towles. *Principles and practices of interconnection networks.* Elsevier, 2004.

[2] Hermann Kopetz. *Real-Time Systems. Design principles for distributed embedded application.* 2nd edition, Springer, 2011.

[3] Thomas H. Cormen, Charles E. Leiserson. *Introduction to algorithms*. MIT Press, 2002.

[4] P. Sankar, C. Chellamuthu, Virtual Clock Scheduling Algorithm for Video Streaming. International Journal of Computer Theory and Engineering. 2009. № 5.

[5] Lixia Zhang. VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks.

[6] Ion Stoica, Hui Zhang, Providing guaranteed services without per flow management.

[7] Hui Zhang, Srinivasan Keshav, Comparison of rate-based service disciplines.

[8] S. Jamaloddin GoIestani. A Stop-and-Go Queueing Framework for Congestion Management.

[9] Hermann Kopetz, Real-Time Systems.Design Principles for DistributedEmbedded Applications. Second Edition, Springer, 2011
[10] Cormen Thomas H., Leiserson, Charles E., Rivest, Ronald L. Introduction to Algorithms 2009.
[11] N. Matveeva, E. Suvorova, Methodology of searching for all shortest routes in NoC. 12th Conference of Open Innovations Association FRUCT. 2012.
[12] A. Eganyan, L. Koblyakova, E. Suvorova, SpaceWire network simulator. SpaceWire-2010. Proceedings of international SpaceWire conference St.Petersburg 2010.
[13] A. Eganyan, E. Suvorova, Y. Sheynin, A. Khakhulin, I. Orlovsky. DCNSimulator – Software Tool for SpaceWire Networks Simulation. International SpaceWire onference 2013, June 2013

Contact Author Email Address

n.matveeva88@gmail.com

Copyright Statement