VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Julius SKIRELIS

INVESTIGATION OF THE VIDEO STREAM CONTROL METHODS

VAIZDO SRAUTŲ VALDYMO METODŲ TYRIMAS

Master Thesis

Informatics Engineering Study Area
Information Electronics Systems, State Code 621E15003
Open Source Systems Specialization

Vilnius, 2016

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

CONFIRMS
Head of Department

____
(*signature*)
Prof Dr Habil R. Martavičius
______________ ____, 2016.

Julius SKIRELIS

INVESTIGATION OF THE VIDEO STREAM CONTROL METHODS

VAIZDO SRAUTŲ VALDYMO METODŲ TYRIMAS

Master Thesis

Informatics Engineering Study Area
Information Electronics Systems, State Code 621E15003
Open Source Systems Specialization

Supervisor	doc. dr. Artūras Serackis	______________	________
	(*scientific title and degree, name, surname*)	(*signature*)	(*date*)
Consultant	dr. Vaida Buivydienė	______________	________
	(*scientific title and degree, name, surname*)	(*signature*)	(*date*)

Vilnius, 2016

Vilnius Gediminas Technical University Faculty of Electronics Department of Electronic Systems	ISBN ISSN Copies No. .... Date ..........-.....-.....

Master Degree Studies **Information Electronics Systems** study programme Master Graduation Thesis

Title	**Investigation of the Video Stream Control Methods**
Author	**Julius Skirelis**
Academic supervisor	**Artūras Serackis**

Thesis language: English

Annotation

The main aim of master`s thesis is to analyze prediction methods, investigate their applicability to predict data bandwidth of the real time video streaming. The aim is achieved by performing analytical review of adaptive video streaming systems, detalizing their advantages and disadvantages. Prediction methods analysis revealed ANN (artificial neural network) features and their applicability on precion of statistical network parameters. Analysis of statistical network parameters acquanted parameters, neccessary to perform optimizations. Concept of adaptive video stream controlled system was developed.

Thesis consists of: 43 pages of text without appendixes, 27 figures, 3 tables, 45 reference entries, 3 appendixes.

Keywords: WebRTC, artificial neural network, adaptive system, intellectual system, open source, NARX neural network, web browser

Vilniaus Gedimino technikos universitetas Elektronikos fakultetas Elektroninių sistemų katedra	ISBN ISSN Egz. sk. .... Data ..........-.....-.....

Antrosios pakopos studijų **Informacinių elektroninių sistemų** programos magistro baigiamasis darbas

Pavadinimas	**Vaizdo srautų valdymo metodų tyrimas**
Autorius	**Julius Skirelis**
Vadovas	**Artūras Serackis**

Kalba: anglų

Anotacija

Magistro baigiamojo darbo tikslas - ištirti prognozavimo metodų galimybes, pritaikant juos duomenų pralaidumui prognozuoti vaizdo transliacijos metu. Iškeltiems uždaviniams įvykdyti buvo atlikta analitinė adaptyviųjų vaizdo transliacijų sistemų apžvalga, kurios metu atskleisti esminiai privalumai ir trūkumai. Prognozavimo metodų tyrimo metu išaiškintos DNT (dirbtinių neuronų tinklų) savybės bei jų pritaikymo galimybės srautų parametrams prognozuoti. Vaizdo transliacijos metu gaunamų statistinių duomenų tyrimas atskleidė parametrus, reikšmingus galimoms optimizacijoms. Atliktas vaizdo srautų valdymo sistemos koncepcijos kūrimas ir jos tyrimas.

Darbo apimtis – 43 psl. teksto be priedų, 27 iliustr., 3 lentelės, 45 bibliografiniai šaltiniai, 3 priedai.

Prasminiai žodžiai: WebRTC, dirbtinis neuronų tinklas, adaptyvi sistema, intelektuali sistema, atvirasis kodas, NARX neuronų tinklas, naršyklė

CONTENTS

Notation and Abbreviations

Mbps	Megabits per second
Kbps	Kilobits per second
REMB	Remote estimated maximum bitrate
NARX	Nonlinear autoregressive neural network with external input
LN	Linear neuron
LRNN	Layer-recurrent neural network
VoIP	Voice over internet protocol
SIP	Session Initiation Protocol
GB	Giga Byte
CPU	Central processing unit
NTP	Network time protocol
Codec	Encoder and Decoder
RTP	Real-time transport protocol
FPS	Frames per second
JSON	Javascript object notation
HTML	Hypertext markup language
RTCP	RTP control protocol

1 INTRODUCTION

Master's thesis examines methods of controlling video streams, possible optimizations on data bandwidth prediction with intelligent systems. To provide a stable video streaming with minimal latency, statistical data should be continuously monitored and encoder parameters adjusted. As base system, open source WebRTC technology was chosen, it provides plugin free real time video and audio conference between browser based environments. Open source technology gives ability to optimize source code, optimizing performance in real conditions. Most scope is given to algorithms and techniques on optimizing bandwidth prediction to keep live stream continuous in extreme conditions.

1.1 Topicality and Aim of the Work

In today's modern mobile technology world, technical environment allows high data bandwidth on mobile devices. Current 4G, 4G+ LTE and upcoming 5G wireless data transfer technologies reach 70 Mbps and more in friendly technical environment. High data bandwidth allows a real time video transfers through mobile devices. However mobile devices do not always work in stable network environment which decreases data bandwidth available dramatically. To maintain such situation and keep data transfer even in worst case scenarios, intelligent software solutions could be applied. The aim of the work is to analyze prediction methods, investigate their applicability to predict data bandwidth of the real time video streaming.

1.2 Tasks of the Work

To maintain aim of the work, three main tasks formulated:

1. Obtain live stream statistical parameters, perform statistical parameters versus time analysis.
2. Chose and analyze three prediction methods suitable to predict statistical parameters of live stream.
3. Propose video stream control system concept based on data channel bandwidth prediction.

1.3 Applied Methods of Investigation and Analysis

To obtain statistical data, solution to log data to text file was chosen because of speed and simplicity. To predict data channel bandwidth three most popular algorithms were selected – NARX, LRNN and classical LN type neural networks. Primary tests were performed in "Matlab" environment. Most suitable algorithm is selected by best performance index value calculated from mean square error.

1.4 Novelty and Practical Value of the Work

Work examines algorithms to be used in real time streams with latency no more than several milliseconds to not to interfere with streaming system. Streaming systems analyzed are strictly plugin free, meaning no additional external software is used, opposing traditional methods using Adobe Flash plugin [1]. Upcoming plugin free technology allows any operating system use its qualities on almost any device. WebRTC technology leaves a gap of optimizations available to be done, this work examines improvements of congestion control for WebRTC technology allowing better network performance characteristics and available bandwidth usage on current and upcoming mobile networks. Optimizations about to deal with packet losses and delays more smoothly allowing continuous stream playback. A concept of video bitrate prediction [19], based on network round trip time and loss parameters was proposed.

1.5 The Scope of the Work

Master`s thesis consists of 6 sections. First section provides introduction, proves topicality and novelty of work, describes tasks and applied methods of investigation.

Second section presents analytical review of adaptive video streaming systems, explains its advantages and disadvantages, reveals scope of open source and closed source systems as well as its techniques of controlling live streams.

Third section consists of analysis on prediction methods suitable to predict data channel bandwidth. Structural schemes of algorithms are provided together with detailed explanations of operation.

Fourth section describes what kind and where from statistical data during live stream is obtained, what filters for data consistency are applied, how encoder parameters changes according to data channel parameters variations. Block diagram of data flow, encoder input parameters analysis are also present in this section.

Fifth section presents conception how intelligent algorithms can be applied to video streaming system, describes practical development of adaptive real time streaming system based on WebRTC open source technology, provides comparison with stock encoder parameters adaptation versus intelligent algorithms results.

In sixth section work is summarized, conclusions presented on applying intelligent algorithms to control real time video streams.

2 ADAPTIVE VIDEO STREAMING SYSTEMS ANALYTICAL REVIEW

Main goal of this section is to review flagship video streaming systems. In the market exists a lot of video streaming systems, this section mostly reviews latest technology – WebRTC. Review of current available products reveals advantages and disadvantages of each, which helps to select proper environment to proceed on with practical part of this work.

2.1 WebRTC

WebRTC – Web based real time communication technology. It does not require any additional plugins or extensions, works in most modern web browsers [19, 20, 21, 22]. Released together with HTML5 standard in 2012. Because of its nature being part of web browser, it is well spread over stationary and mobile devices. Every “Google Chrome”, “Mozilla Firefox” or “Opera” browser in mobile phone, computer or TV [2] support it. HTML5 standard release many other extensions, like capturing video and audio streams directly from camera, “Websockets” technology for direct socket connections between browsers. These extensions in stack allowed WebRTC to be implemented.

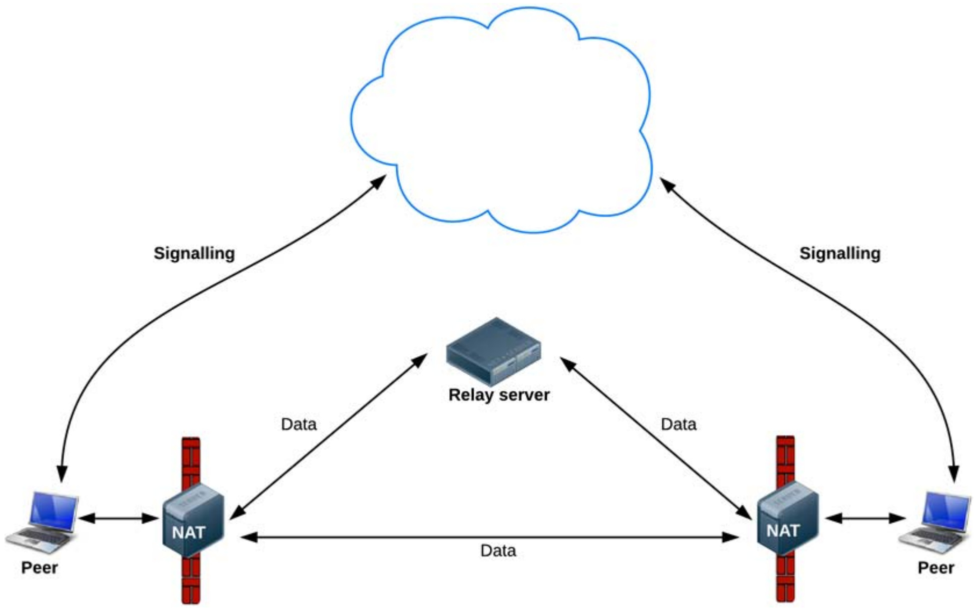


Figure 2.1. WebRTC connection diagram [3]

Figure 2.1 represents connection diagram of WebRTC connection. Signaling server serves to exchange connection data and peer states, peers opens UDP ports (*Hole punch* [4] technique) and direct data channel between peers is established. If peer locates behind NAT firewall, relay server with external public IP addresses should be used to ensure connection between peers is established in 99% probability.

Peer – in WebRTC environment is an end node, which has browser with WebRTC and HTML5 frameworks available. Not only a browser can be an endpoint, browser environment can be simulated with any programming interface, including chromium sources into it. One of the most popular environments for deploying endpoint or signaling server – Node.js environment [3].

WebRTC developer provides example applications, deploying peer client and server. Examples contains chromium framework sources, extended to C++ programming language. Technology seamlessly controls operations of TCP/IP layers [3], without need of user intrusion, however native application, which is open source, gives ability to see all main operations and techniques operating network management and codec control, and also recode it.

Technology has two type of connections, which can be parallelized: Media channel [3] and Data channel [5]. Media channel is strictly for media type data transfer, video and audio, buffers are controlled automatically. Data channel is dedicated for text data transmission, plain text. It can be used for transferring some direct messages between peers, allowing real time chat, or even file transfer may be established. Combination of those two can manage complete real time video conference system with chat and file transfer.

Advantages:

- Open source
- Uses open source VP8 codec
- Has documentation for environment preparation and binary deployment
- Easily integrates into other programming environments like Java, Node.js
- Has full support on Windows, Linux, Android and partial support for iOS operating systems
- Uses direct UDP connection between peers – no need for transparent server

Disadvantages:

- Source files use a lot of storage space, because of chromium sources (~15GB)
- Compiles using Google`s own compiler (ninja)
- Not stable, technology is in development state
- Proprietary h.264 codec performs better in comparison with VP8
- Source code is hard to read because of conjunctions to chromium sources
- Currently only supports two endpoints at once

Some researches [6], [7] already analyzes additional techniques on optimizing WebRTC with adaptive methods, one of most interesting is “SPRAY” protocol [6] which denotes ability to use WebRTC technology for distributed applications. Protocol explains how effectively sample each of peer without incorporating classic star topology where master and slaves are active, but to use signaling feature of each node to transfer messages about node state, thus allowing node joins, shuffling, handles endpoint crashes and disconnects.

WebRTC technology is already widely used to transfer various live media streams, including sports championships [7], other techniques were proposed as alternative to WebRTC for such cases, because WebRTC technology is not able to fulfill such big capacity or network streams (reaches 3.5Mbps on each node) correctly, leading to constant stream connection failures.

Most of IP television providers researches WebRTC as technology of their future [2], leading to complete elimination of IPTV transcoders. They about to rely on TV connected to network, which exists on provider`s network to transfer media streams without any additional hardware required. It would also let bring down the costs of provider and end-user itself, making system fully automated, acting as a web page with great graphical interface [18, 19]. However, significant security methods have to be applied to prevent cheating and hacking of system [31].

2.2 Kurento media server

Kurento is a media server solution, which involves all modern technologies in one package. It includes "GStreamer" [8], has VP8, H.264 and H.263 codecs, uses pure HTTP, RTP and WebRTC environments. Usage of native WebRTC sources gives all advantages mentioned in section 2.1 , plus advantages given with Kurento product itself.

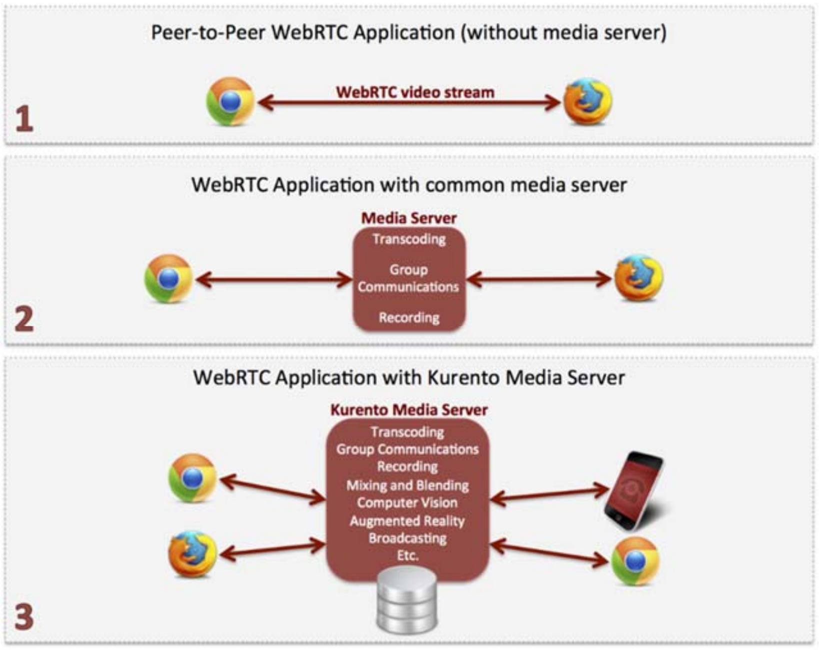


Figure 2.2. Comparison of pure peer-to-peer connection versus media server [https://www.kurento.org]

Three main conceptions (Figure 2.2) are known for use with WebRTC technology: 1 – direct peer-to-peer connection between endpoints, no intermediate server is used, no media infrastructure; 2 – simple intermediate media server is used, which acts as an endpoint for both of external endpoints, performs as stream router allowing introduce techniques like record of media stream, additional transcoding; 3 – high level media server, allowing additional stream mixing, additional software (for example GStreamer [8]) is usually used to perform media transformation, but is rather complex and resource demanding service.

Advantages:

- Open source
- Requires difficult media server to be installed
- Supports one-to-many connections

Disadvantages:

- A lot of code changes every update makes it inconvenient to deploy applications
- Requires Kurento framework to be included into all deployable applications

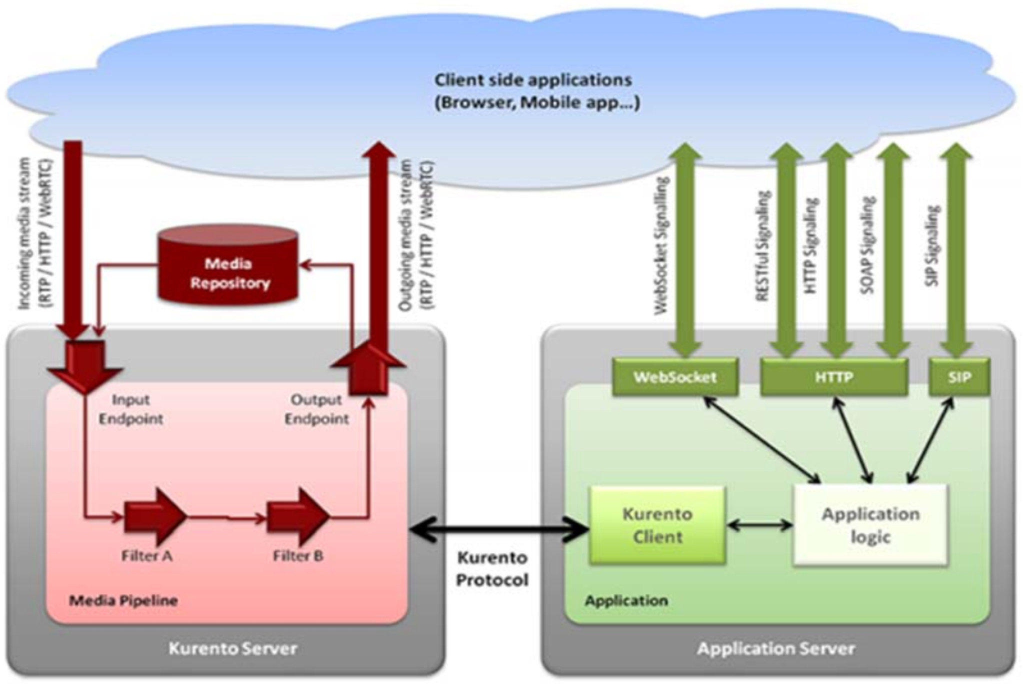


Figure 2.3. Kurento media server architecture

Kurento media server structure is shared between three nodes (Figure 2.3): Kurento cloud server; application server on user managed server; end user application. Application server is mostly closed source, interchanges data with Kurento protocol [13]. User application has connection between Kurento server and application server. Such architecture limits any source code changes by user, but offers great features in media server inside. One of the most feature is that Kurento media server can be used as a bridge between different codecs on endpoints: VP8 encoded endpoint <-> Kurento media server <-> h.264 [40, 41] decoded endpoint.

2.3 OpenTok

OpenTok is another WebRTC based commercial product, it originally was launched using obsolete Adobe Flash technology [9]. Current release offers complete embedded communication platform, standalone, or cloud based. Platform is highly customizable, however core framework is closed source and only external functions are customizable.

Platform brings WebRTC features of media streaming (video and audio), additionally screen sharing, multi connection conference, plugin for Internet Explorer browser (which natively does not support WebRTC connections) features are available.

OpenTok documentation describes additional stream encryption [9], additional software firewall feature for increasing security concerns. Android and iOS operating systems are supported – additional application must be installed to provide full feature set.

Advantages:

- More stable than pure WebRTC
- Constant updates
- API for external applications

Disadvantages:

- Partially closed source
- Very high peripheral resources consumption
- Requires very complex media server installation

OpenTok software took its niche mostly on making various presentations, remote studies. One conference room is limited to 20 participants [10], has a great user friendly graphical interface, connected to most popular social networks.

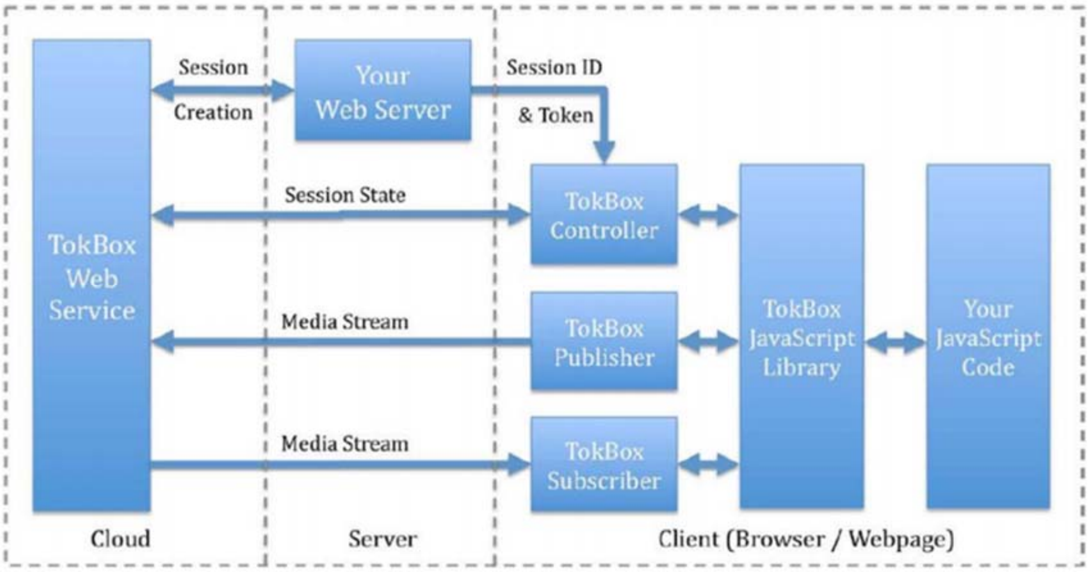


Figure 2.4. OpenTok structure diagram [10]

OpenTok software is not complete deployable package, it is hardly tied to cloud services provided by developer (Figure 2.4). Software allow modifying final blocks of software, but main protocols remain closed sourced.

2.4 Skype

Skype software is a closed source product. Skype is one of the most popular VoIP tool, which includes video and audio stream through peer-to-peer network connection, its specifications remains unknown. Only partial network architecture diagram (Figure 2.5) is available, where network is split to two types of nodes: supernode – node which accepts external connections, may be skype server, or user with external IP address; node – user device which does not serve as an signaling server, passes data through peer-to-peer connection. Network connection topology is very similar to WebRTC technology described in section 2.1 .

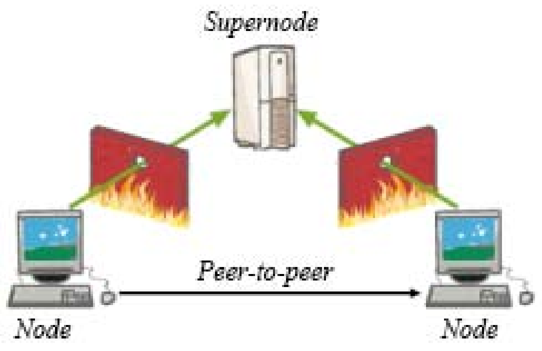


Figure 2.5. Skype network architecture [11]

Skype software was taken into review because it accomplishes network connections in same manner as WebRTC does. Skype also support data and media channels, for low resolution video encoding VP8 open source codec is used, for high resolution – proprietary h.264 codec [45, 43, 42]. Ability to switch codecs allows better data flow in Skype software rather than WebRTC technology (because of better compression and higher peripheral resources load rates). Security concerns and congestion control technologies are not publicly explained, however experimental tests shows that Skype software has great algorithms and/or intelligent systems applied.

Advantages:

- 99% probability of connection to be established
- Optimized algorithms allow adaption of stream parameters on network environment changes
- Switches between codecs allows best bandwidth-resources load ratio

Disadvantages:

- Closed source, no protocol and security mechanisms documented

- License prohibits usage for commercial purposes, unless "Skype business" license applied.

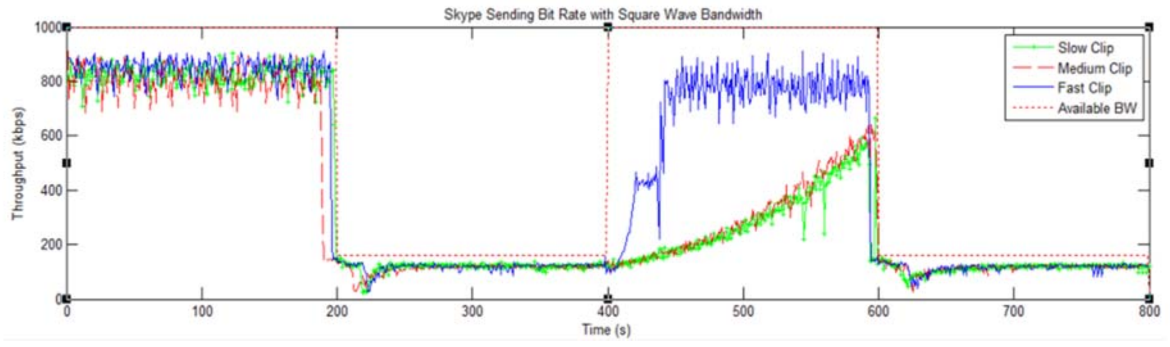


Figure 2.6. Skype software adaptive mechanism [12]

Figure 2.6 represents bandwidth usage of Skype software where sudden network throughput is simulated, rapid drop of bandwidth seem to cause internal adaptation algorithms to slightly undershoot, but connection rapidly stabilizes and stream continues. After limiter switched off, adaptive algorithms start to recover situation, smoothly increasing usage consuming. Since algorithms applied in Skype software are proprietary, it is unknown what techniques are used, but overall results are excellent, in both, natural fluctuation and sudden drop situations emulated.

2.5 Section generalization

Review of adaptive streaming systems shows that WebRTC technology has no optimized techniques to properly perform congestion control, those are later applied by vendors of final products. WebRTC itself presents well done connection establishment mechanism, developers of subsidiary products improve it even more by implementing external interfaces like separate servers for UDP hole punching. To perform media tasks, intermediate media servers are applied instead of direct connection between endnotes [36], it grants full control of media streams rather than performing tasks in user accessible browser interface.

On all occasions external signaling server is used for route SIP messages between peers, interchange of internal and external IP addresses, other connection related messages [37] before and after direct peer-to-peer connection established.

Review of Skype software, acknowledge it uses same connection topology as WebRTC does, shows that WebRTC is just inclusion of well know topology into new environment – modern web browsers directly.

3 ANALYSIS OF PREDICTION METHODS

Real time video stream encounters with network connection instability, which is expressed as sharp variation of network parameters: round trip time value measured in milliseconds; lost packets count; received frames count. Parameters tend to be non-periodic instantaneous values. To accomplish prediction of bandwidth available with minimal latency, artificial neural network could be used. Current section represents three intelligent prediction methods to predict values.

Artificial neuron network is basically a mathematical representation of human body neural network. The main reason neural network solutions are selected for this work is a nonlinear features of neural networks and a simple implementation. Basically all main neural network methods share same features – there is an input, and an output. Between input and output stages each input value has a weight coefficient, and products of those are accumulated. Lately summarized value is passed to linear, or nonlinear activation function which produces the answer. Additional techniques like input values, intermediate summarized values or output values may be fed back to input with delay in samples. Artificial neural network may consist of numerous amount of intermediate layers, various combinations (serial, parallel, mixed) [29], although increasing complexity and resources needed to accomplish computations. Usually increasing system complexity increases accuracy, there exist point when neuron count reaches maximum where accuracy increase is negligible in comparison with system complexity increase rate. To test accuracy of each neural network, prediction of random jitter data was performed.

3.1 Linear neuron (LN)

Linear neuron is the basic element. For system to work, neuron should be trained. Train is a step where weight coefficients are found. Training operation is performed by having an item set which contains matching pairs of inputs and desired outputs. Initially those coefficients are randomly generated [. While performing train state, loop iterates over neural network adjusting coefficients so it matches known values with certain error. For this to accomplish, after each iteration calculated, output value is compared to desired output and error is calculated. Error value may be subtraction product directly, mean square error value, gradient distribution or any other type, depending of desired accuracy of training. Training operation is flagged as finished upon set conditions (error or distribution threshold) acquired.

For very small error values, or impossible combination set it can take very high or in some cases even infinite loop iterations to be done, so training operation is usually also narrowed by setting iteration count threshold, creating race condition on which event happens earlier.

$$y = kx \tag{3.1}$$

Linear activation function is a slope (equation 3.1) which displaces sum product of inputs, intended to adjust output on final stage of multilayer neural network.

Coefficients calculated during training state is a part of system and should be transferred together with whole system deploy package. Such system may be used not only for prediction, but also for clustering or binary output (decision).

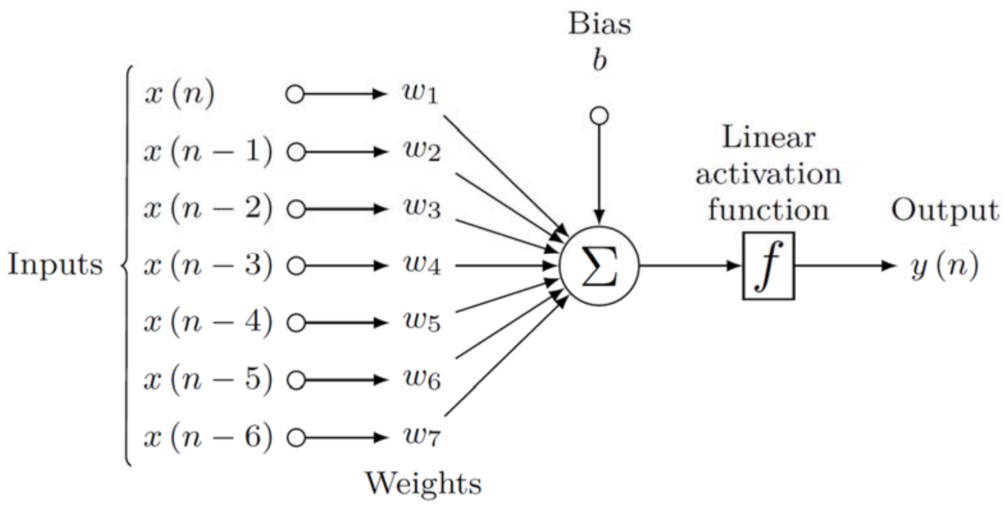


Figure 3.1. Structure diagram of linear neuron

Figure 3.1 shows internal structure of linear neuron. Example describes one input variable, which has delay of 6 samples. System can be configured to more inputs (they will influence the output as well) with their weight coefficients.

$$y(n) = f\left(\sum_{j=1}^{7} w_j x_j\right) \tag{3.2}$$

Equation 3.2 represents mathematical expression of linear neuron, where f - activation function; j - count of inputs, whether it is current input, or delayed input; w_j - weight coefficient for input j; x_j - input value; $y(n)$ - output of neuron for input sample n.

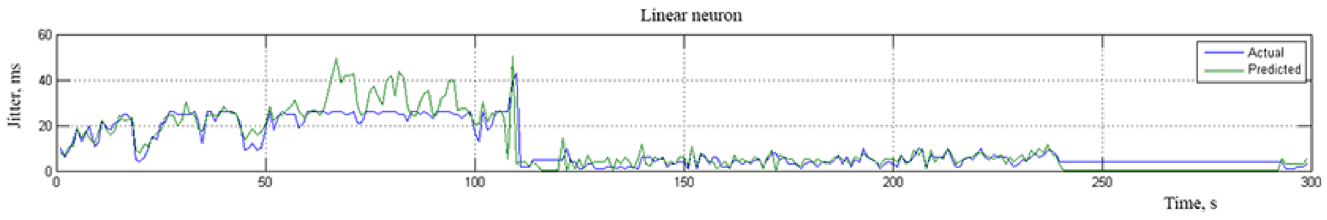


Figure 3.2. Prediction of random jitter values using LN predictor

Figure 3.2 illustrates results of predicting network jitter value using linear neuron network. Direct comparison of random generated and predicted signal curves on single graph shows slight overshoot of predicted values, however signal trend is maintained.

3.2 Layer-recurrent neural network (LRNN)

Recurrent neural network is a type of neural network, similar to linear neuron, but has a feedback of internal calculation steps to input of network. It allows to apply set of weights recursively [26]. Such types of neural networks are widely used in speech recognition and other language applicable models because of its ability to have infinite dynamic response versus time series [23].

Most of the features of linear neuron are inherited. Usually hidden layers has nonlinear activation functions which increases computational resources needed to perform arithmetical calculations, since nonlinear activation function may require exponential or trigonometrical functions to be calculated. Effective training of recurrent neural network is not an easy task, because of unstable relationship between parameters [25].

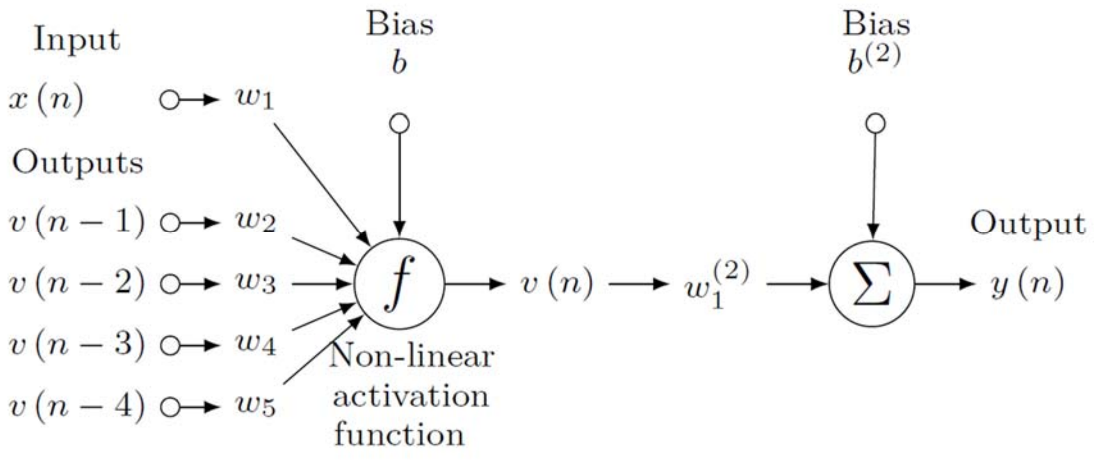


Figure 3.3. Structure diagram of LRNN

Example of layer recurrent neural network structure diagram is given in Figure 3.3. Network model basically consist of two neural networks, from where they conjugate, is the feedback node. LRNN network can be used with multiple input variables. Network is stated as sequential network because of feedback and time based delays, this feature makes LRNN network to be fault-tolerant.

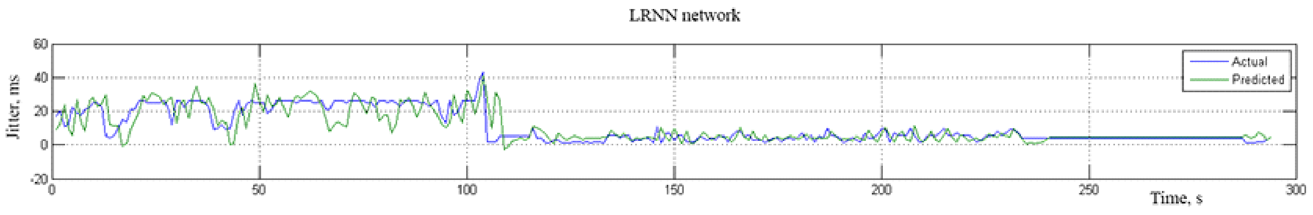


Figure 3.4. Prediction of random jitter values using LRNN predictor

Figure 3.4 illustrates prediction of jitter value. Prediction result shows slight response delay and undershoots introduced, better accuracy in comparison with linear neuron predictor. Predictor is configured to have one input and one output, two sample feedback from hidden layer with 2 neurons.

3.3 Nonlinear autoregressive neural network with external input (NARX)

NARX neural network model is particularly suited for modeling nonlinear systems which depends on time series. It features gradient-descending learning algorithm, because of that model converges and outperforms other neural networks. NARX model is suitable to predict long time because of delayed output feedback which allows back propagation through time.

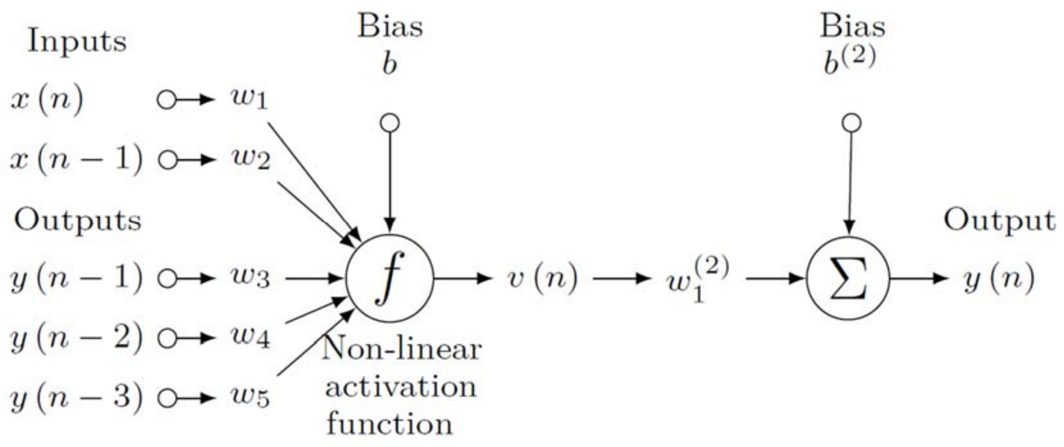


Figure 3.5. Structure diagram of NARX

Figure 3.5 shows internal structure of NARX network (hidden layers are not shown), mathematical representation of model can be denoted as:

$$y(n) = F(y(n-1), y(n-2), \ldots x(n), x(n-1), x(n-2), \ldots) + err(n), \quad \textbf{(3.3)}$$

where $y(n)$ is current output of predictor, $x(n)$ is current input, $err(n)$ is error term, F is nonlinear activation function. Error term here denotes a difference between actual value and predicted value, since they may not be exact in value.

Network is trained taking into computations real output of network which leads to less error gradient and great converge times. However, network can be “over trained” leading to false prediction instead of better forecasts. NARX model tend to have good performance ratios, because of this, weight coefficients can be small in magnitude [26], prediction response is smooth.

Network can be trained using function to update weights based on Levenberg-Marquardt optimization or Bayesian regularization techniques, it minimizes squared error and weight values. Depending on network size, one or other optimization function can produce better results. Because of optimizations to be performed, and nonlinear activation functions used, input values should be kept as small as possible, normalization should be applied.

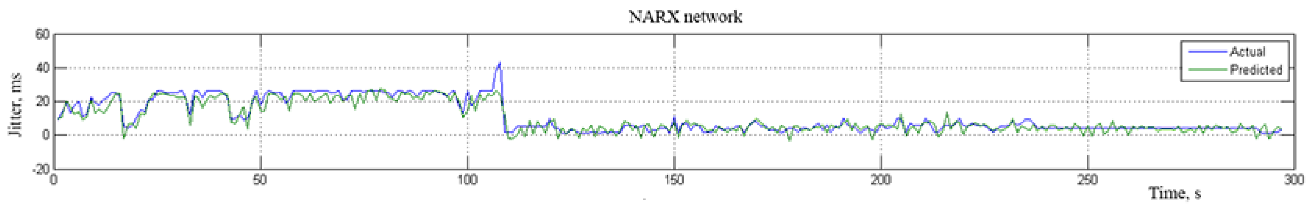


Figure 3.6. Prediction of random jitter values using NARX predictor

Figure 3.6 illustrates results of NARX predictor, accuracy of predicted values is best in comparison with LN and LRNN networks, predicting random generated jitter values of network connection. Predictor is configured to one external input with two delay samples of input, as well as two delay feedback samples of predictor output.

3.4 Section generalization

Analysis showed that NARX predictor has best performance index in comparison to linear neuron and LRNN predictor. Prediction of frames with NARX predictor faces with the mean square error near 11, which means number of frames could be predicted with an average error of 3-4 frames. This result leads to select NARX predictor to be used in encoder parameters prediction later in this work.

Analysis showed that neural networks with backpropagation tend to pass their output forward through network, as well as propagating the error backwards. Main idea of backpropagation is to reduce the errors while network *learns* the training data, as training procedure starts with random generated weight coefficients, main goal is to adjust weights so the error would be smallest in magnitude.

4 VIDEO STREAM STATISTICAL DATA SCREENING

This section describes what statistical data is used to control video streams, how particular values are obtained. Since work analyzes open source WebRTC technology, only its particular statistical data and parameters are discussed.

To perform the aim of the work, necessary statistical data should be obtained. Media streams are based on many parameters, however not all of them are applicable for purposes to be applied to artificial neural networks (examined in chapter 3). Data output capture techniques are discussed in this chapter, determining whether collectable data is suitable for further analysis, also its origins and sense in perspective of determining current state of network environment as well as media encoder itself.

4.1 WEB browser

To predict network connection changes and adjust encoder parameters, statistical data of network loss rate, round trip time, jitter and frames received values should be collected. Since version 23, Google Chrome supports WebRTC framework by default, it gives ability to test and deploy application on real system.

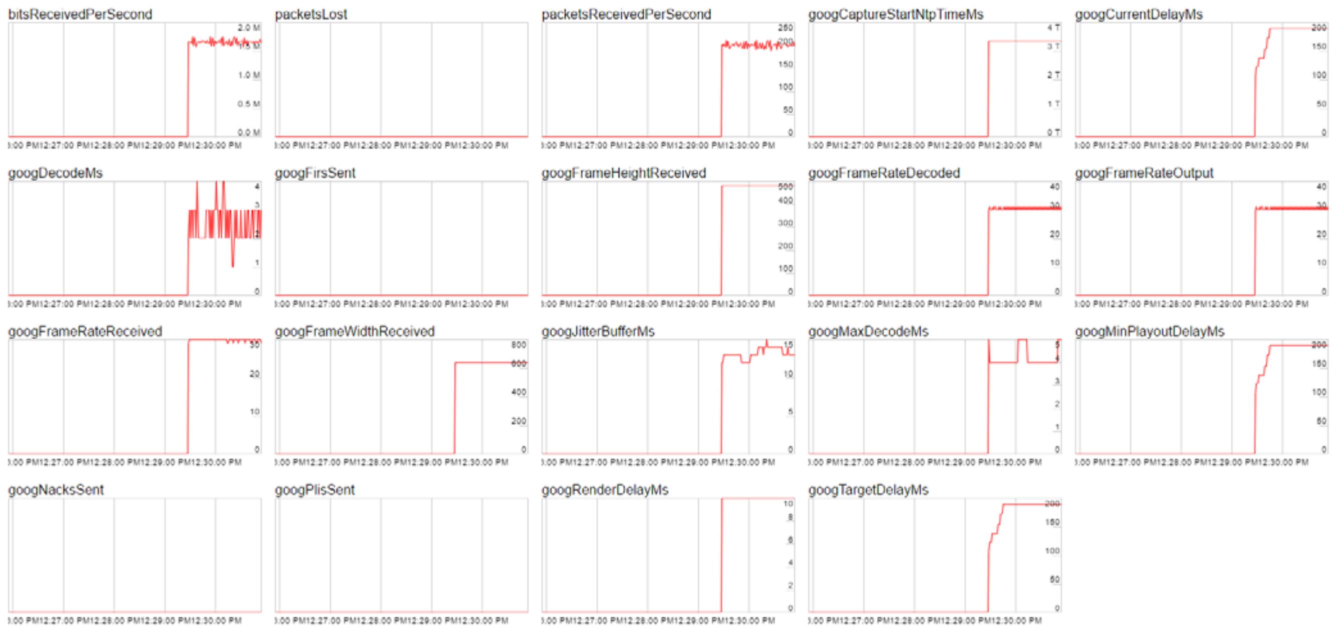


Figure 4.1. Google Chrome outgoing stream statistical data

Figure 4.1 represents outgoing statistical data collected in Google Chrome browser. Data is being shown graphically which gives ability to view state of media stream graphically. It is known [13] that VP8 codec has its own predictors for adjusting macroblock size [13], so basically adequate parameters should be adjuster for codec to achieve best results. Mainly all those parameters are visible in Figure 4.1 which represents statistical data collected over short media stream.

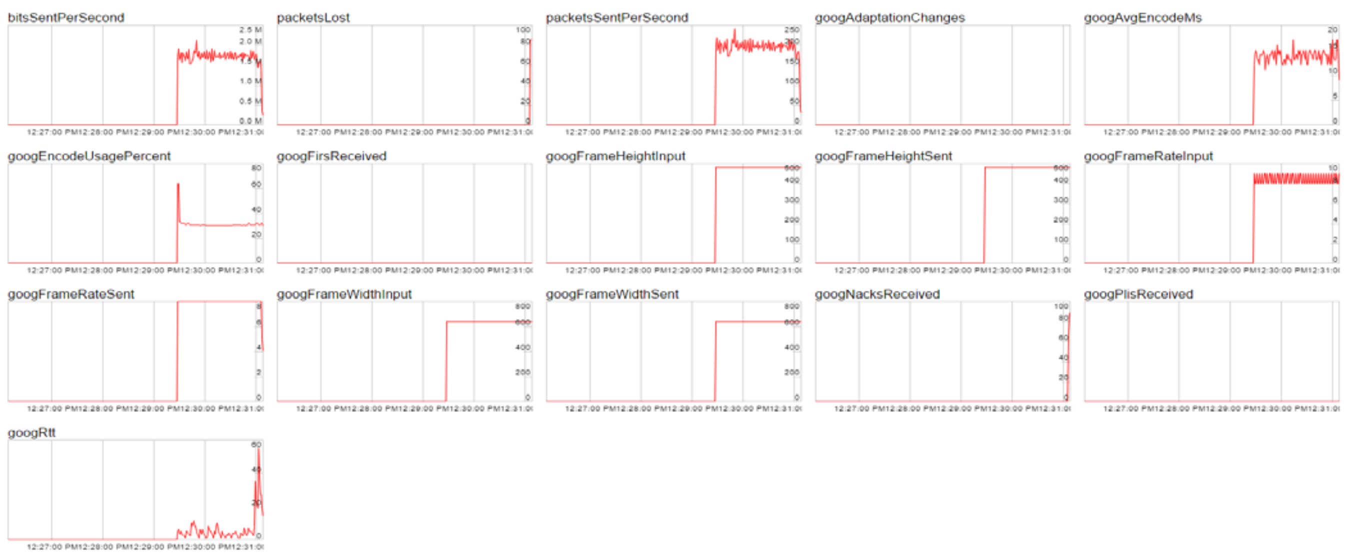
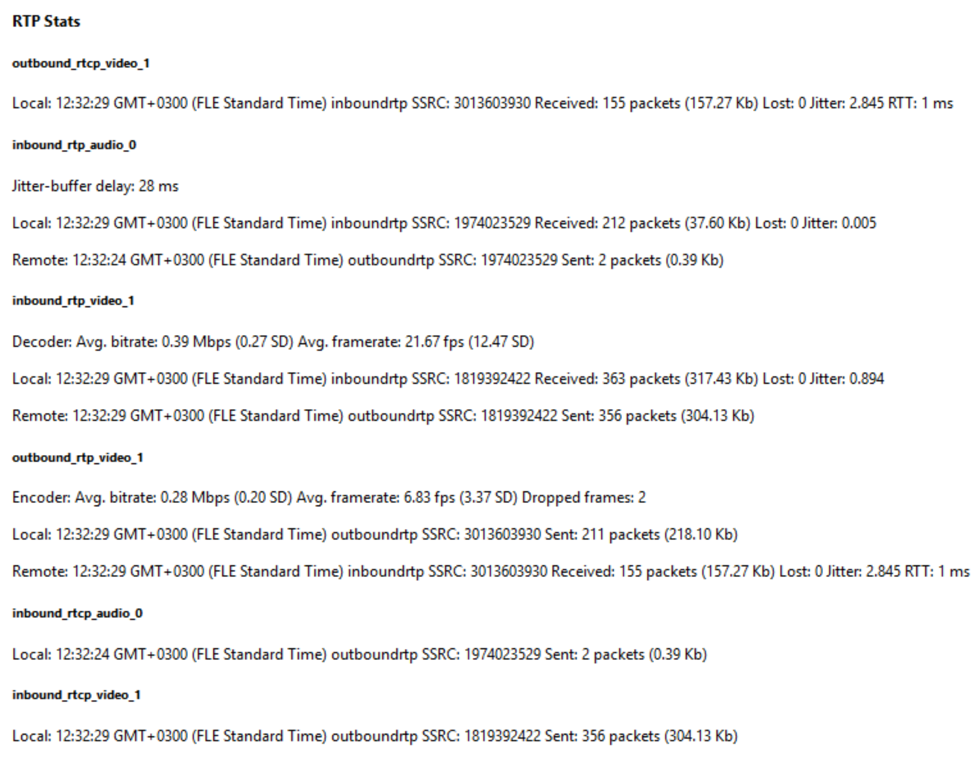


Figure 4.2. Google Chrome incoming stream statistical data

Figure 4.2 respectively shows incoming statistical parameters collected on remote endpoints. Figure clearly shows different parameters which describes tendency that encoder and decoder uses different parameters to configure itself.

For easy display of media streams HTML5 standard includes plugin container (media player). It is inserted into webpage using `<video>` tag. Stream source from web camera is obtained via JavaScript programming environment.

Google Chrome WebRTC statistics page also allows to save statistical data log to file, however it is JSON [14] structure encoded, to decode collected data into data arrays, simple JavaScript parser is made.



RTP Stats

outbound_rtcp_video_1

Local: 12:32:29 GMT+0300 (FLE Standard Time) inboundrtp SSRC: 3013603930 Received: 155 packets (157.27 Kb) Lost: 0 Jitter: 2.845 RTT: 1 ms

inbound_rtp_audio_0

Jitter-buffer delay: 28 ms

Local: 12:32:29 GMT+0300 (FLE Standard Time) inboundrtp SSRC: 1974023529 Received: 212 packets (37.60 Kb) Lost: 0 Jitter: 0.005

Remote: 12:32:24 GMT+0300 (FLE Standard Time) outboundrtp SSRC: 1974023529 Sent: 2 packets (0.39 Kb)

inbound_rtp_video_1

Decoder: Avg. bitrate: 0.39 Mbps (0.27 SD) Avg. framerate: 21.67 fps (12.47 SD)

Local: 12:32:29 GMT+0300 (FLE Standard Time) inboundrtp SSRC: 1819392422 Received: 363 packets (317.43 Kb) Lost: 0 Jitter: 0.894

Remote: 12:32:29 GMT+0300 (FLE Standard Time) outboundrtp SSRC: 1819392422 Sent: 356 packets (304.13 Kb)

outbound_rtp_video_1

Encoder: Avg. bitrate: 0.28 Mbps (0.20 SD) Avg. framerate: 6.83 fps (3.37 SD) Dropped frames: 2

Local: 12:32:29 GMT+0300 (FLE Standard Time) outboundrtp SSRC: 3013603930 Sent: 211 packets (218.10 Kb)

Remote: 12:32:29 GMT+0300 (FLE Standard Time) inboundrtp SSRC: 3013603930 Received: 155 packets (157.27 Kb) Lost: 0 Jitter: 2.845 RTT: 1 ms

inbound_rtcp_audio_0

Local: 12:32:24 GMT+0300 (FLE Standard Time) outboundrtp SSRC: 1974023529 Sent: 2 packets (0.39 Kb)

inbound_rtcp_video_1

Local: 12:32:29 GMT+0300 (FLE Standard Time) outboundrtp SSRC: 1819392422 Sent: 356 packets (304.13 Kb)

Figure 4.3. Mozilla Firefox statistical data

Figure 4.3 shows average statistical data collected on Mozilla Firefox using same environment, it is clearly seen that Mozilla Firefox displays average values instead of instantaneous, no graphs are provided, however data describing state of codec is still available.

Statistical data collection from web browsers showed that different browsers use same data to configure VP8 codec. Most meaningful data for end user is FPS (frames per second), this parameter describes smoothness of video stream. Experiments shown framerate is widely limited by hardware camera parameters itself. However, to improve issues with media stream shudder, low level parameters should be taken into account. Experiment clearly states the issue with those parameters – they do not reach to network parameter changes instantaneously. Few seconds to few tenth of seconds delay is perceptible for parameters on sudden packet loss, which leads into complete stream cease in timeframe of delay. Such cases make it nearly impossible to use this system on mobile networks when mobile device is moving across the area with poor signal quality.

Parameters like number of received frames, RTT (round trip time), lost packets rate and jitter were selected for further investigation. Round trip time is a parameter which mostly describes network delay, is a direct parameter describing time taken for a single packet to be sent and answer to be received from remote endpoint. If a network delay increases, round trip time parameter also increases respectively, and such case is clearly visible on experiment where network delay is increased artificially. Lost packets value describes a ratio of packets lost during a sampling time interval (which is a 1 second by default), it denotes an unstable network connection, changes between mobile network stations and similar. Jitter parameter describes a natural phenomenon of packets to be sent and received in non-exact same time intervals, but sudden increase in value would sign instability in network connection which should be handled by codec accordingly.

4.2 Native WebRTC application

Deployed web browser software only allows to log statistical data, but does not allow to access any data buffers or parameters itself. To obtain access to methods controlling encoder, low level software has to be used. As WebRTC technology is open source, whole Chromium (Google Chrome framework) is publicly accessible, those sources about to be used as a base for direct statistical data capturing and concept of adaptive streaming system development environment.

After checkout of official WebRTC repository, 15GB of source files in Visual Studio project are received, all perquisites documented in installation instructions performed, deploy package is compiled. Package contains standalone application for signaling server, which is used for connection data interchange (SIP packets) and perform conference rooms management.

After examination of the code, preparing UML diagrams (Appendix Figure 1 and Figure 2), mechanisms of configuring encoder parameters were acknowledged. Analysis showed, that encoder parameters basically depends on three variables [15]:

1. CPU overuse. Detection implemented on comparing input framerate to encoding framerate, if encoding framerate is much less than input framerate, CPU overuse flag is raised meaning CPU cannot handle such high framerates. Closed loop frame drops are then initialized, if CPU overuse condition still persists – input video resolution is downscaled until encoding rate reaches stable state.
2. QP (quantization parameter). Mechanism detects QP value, if its value is being very high, it is assumed video quality is low (as well as bitrate) and video is believed to look blocky (macroblocks are big). Figure 4.4 represents dependence of bitrate versus quantization parameter. If high QP value condition is met video downscaling before encoding is initiated.

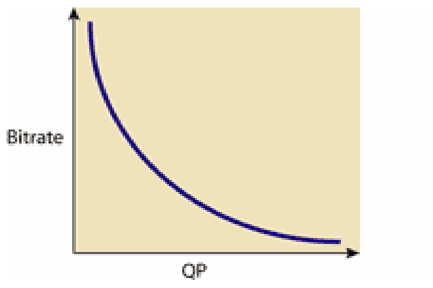
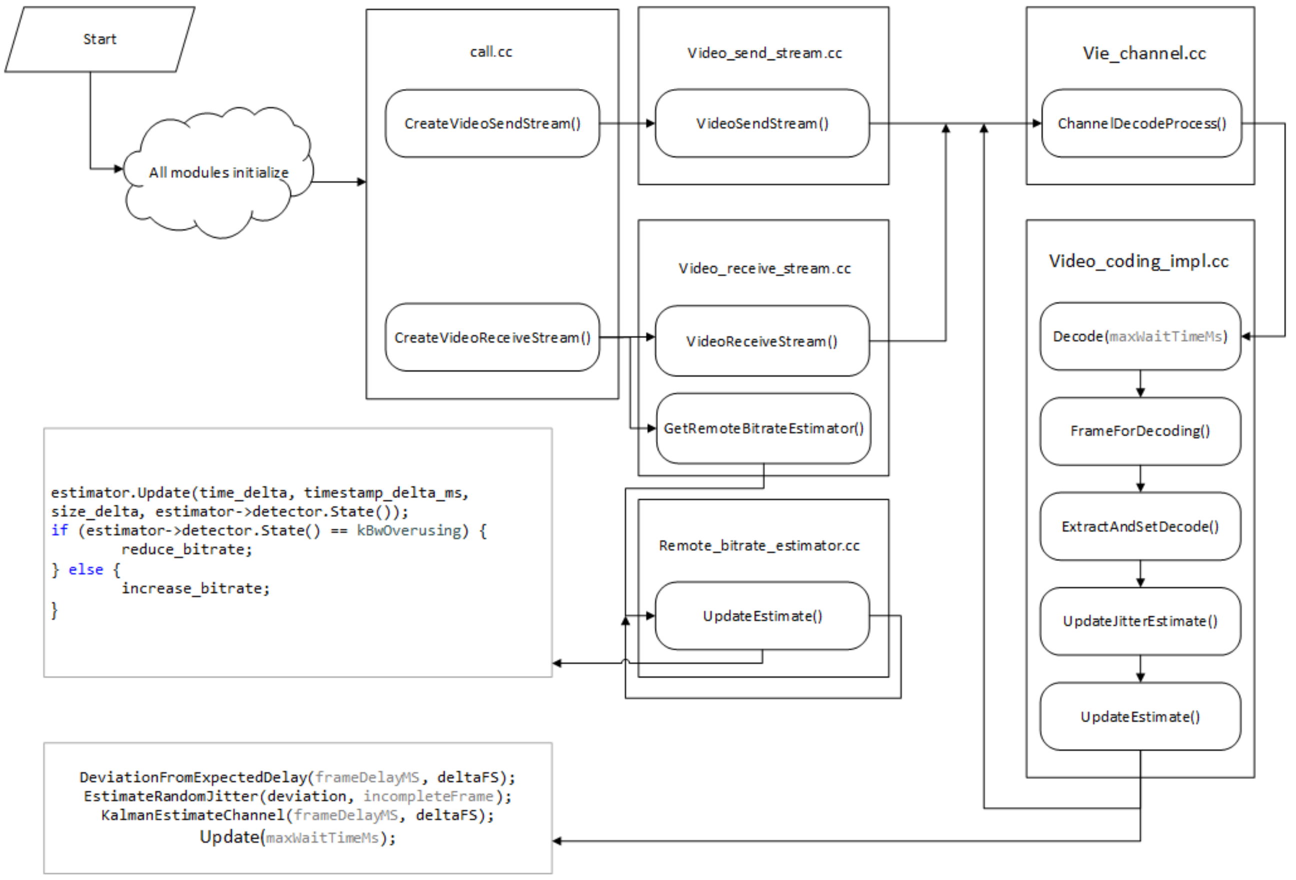


Figure 4.4. QP value versus bitrate

3. Estimated bitrate. Bitrate estimation is implemented by relying on REMB feedback scheme of RTP packets. This value is calculated by sharing synchronized packets of current time (NTP protocol) and applying arrival-time model in remote endpoint. Model states that each received frame is assigned to variable $t(i)$ which denotes time when whole frame is received. Group of packets with same timestamps is denoted as $T(i)$. Packet losses are not taken into account. Frame stated as delayed if equation $t(i)-t(i-1)>T(i)-T(i-1)$ is satisfied, meaning arrival time difference is higher in value in comparison with timestamp difference. Relative arrival time is denoted as $d(i) = t(i)-t(i-1)-(T(i)-T(i-1))$, and applying Kalman filter estimation [16] according to relative arrival time and its reference to NTP received timestamp bitrate is estimated. Local endpoint receiving REMB value from endpoint adjusts encoder parameters.

Experimental analysis by adjusting network parameters manually (increasing loss rate, jitter and limiting bandwidth) showed that first two encoder adjustment techniques works fine, while adjustment by remotely estimated bitrate does not work correctly in many cases, causing total connection loss because of inadequate encoder parameters adapted.



```
estimator.Update(time_delta, timestamp_delta_ms,
size_delta, estimator->detector.State());
if (estimator->detector.State() == kBwOverusing) {
        reduce_bitrate;
} else {
        increase_bitrate;
}
```

```
DeviationFromExpectedDelay(frameDelayMS, deltaFS);
EstimateRandomJitter(deviation, incompleteFrame);
KalmanEstimateChannel(frameDelayMS, deltaFS);
Update(maxWaitTimeMs);
```

Figure 4.5. WebRTC adaptive logic sequence

Figure 4.5 represents simplified login sequence executed in loop every packet received during media streaming process. Most sensitive part of adjusting encoder by REMB received bitrate is handled in function “UpdateEstimate()”. Issue leading to complete stream loss is caused because of encoder is adjusted right after packet messaging about available bitrate from remote endpoint is received. Such event happens no more often than set interval $t\_min\_fb\_interval$, and no less often than interval $t\_max\_fb\_interval$, if no such packet is received during time interval two times the $t\_max\_fb\_interval$, algorithm assumes such packet as lost and will result in a halving the send rate and trying to send out packets without readjusting the encoder.

Source code 4.1. UpdateEstimate() contents

```
if (last_fraction_loss_ <= 5) {
      // Loss < 2%: Increase rate by 8% of the min bitrate in the last
      // kBweIncreaseIntervalMs.
      // Note that by remembering the bitrate over the last second one can
      // rampup up one second faster than if only allowed to start ramping
      // at 8% per second rate now. E.g.:
      //   If sending a constant 100kbps it can rampup immediatly to 108kbps
      //   whenever a receiver report is received with lower packet loss.
```

```
  //   If instead one would do: bitrate_ *= 1.08^(delta time), it would
  //   take over one second since the lower packet loss to achieve 108kbps.
  bitrate_ = static_cast<uint32_t>(
      min_bitrate_history_.front().second * 1.08 + 0.5);

  // Add 1 kbps extra, just to make sure that we do not get stuck
  // (gives a little extra increase at low rates, negligible at higher
  // rates).
  bitrate_ += 1000;

  if (event_log_) {
    event_log_->LogBwePacketLossEvent(
        bitrate_, last_fraction_loss_,
        expected_packets_since_last_loss_update_);
  }
 } else if (last_fraction_loss_ <= 26) {
  // Loss between 2% - 10%: Do nothing.
 } else {
  // Loss > 10%: Limit the rate decreases to once a kBweDecreaseIntervalMs +
  // rtt.
  if (!has_decreased_since_last_fraction_loss_ &&
      (now_ms - time_last_decrease_ms_) >=
          (kBweDecreaseIntervalMs + last_round_trip_time_ms_)) {
    time_last_decrease_ms_ = now_ms;

    // Reduce rate:
    //   newRate = rate * (1 - 0.5*lossRate);
    //   where packetLoss = 256*lossRate;
    bitrate_ = static_cast<uint32_t>(
        (bitrate_ * static_cast<double>(512 - last_fraction_loss_)) /
        512.0);
    has_decreased_since_last_fraction_loss_ = true;
  }
  if (event_log_) {
    event_log_->LogBwePacketLossEvent(
        bitrate_, last_fraction_loss_,
        expected_packets_since_last_loss_update_);
  }
 }
}
```

Source code 4.1 represents bitrate adjusting mechanism denoting dependence of bitrate passed to encoder versus loss. Algorithm about to work smoothly decreasing or increasing bitrate, it is denoted as:

- If less than 2% of packets are lost, meaning loss is very low, algorithm tries to increase bitrate by 8% by every iteration of loop
- If packet loss is in range of 2-10% no adjustment is done – REMB received value is passed directly to encoder and updating bitrate buffer accordingly
- If packet loss is greater than 10%, algorithm starts to decrease bandwidth available value by nonlinear value calculated by taking into account RTT variable. Such algorithm assumed to decrease available bandwidth more rapidly maintaining active media stream connection.

Source code allow to adjust bandwidth increment and decrement rates (relative to NTP time) with default values denoted in Source code 4.2.

Source code 4.2. Adaptation intervals and bitrate thresholds

```
const int64_t kBweIncreaseIntervalMs = 300;
const int64_t kBweDecreaseIntervalMs = 1000;
const int kDefaultMinBitrateBps = 1000;
const int kDefaultMaxBitrateBps = 1000000000;
```

During experiment various combinations of rates were tested but no significant increase in overall system quality automatically managing sudden network parameter changes was achieved. Results states that such algorithm to decrease bandwidth by iterating through loop after a packet is received adapts too slowly before packet timeout flag is raised and stream is terminated making such system nearly impossible to use on mobile devices with mobile network enabled.

To perform simulation of situations where algorithm is not capable to manage extreme situations data screening of various parameters was implemented. To capture statistical data, method to log variables to text file was chosen, tab separation between each sample values was selected because of easy further import to any simulation system.

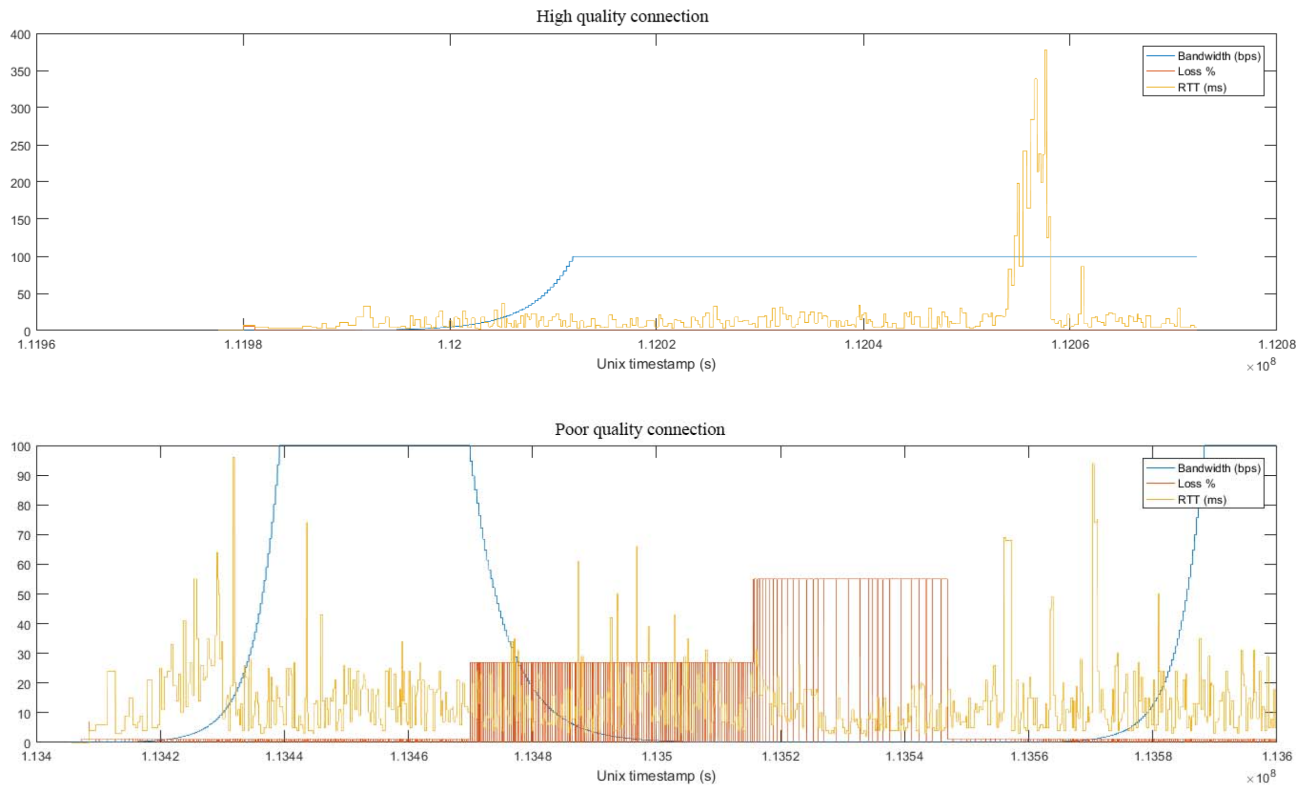


Figure 4.6. Bandwidth, loss rate and RTT values on high and poor quality connections

Figure 4.6 shows how adapted bandwidth (blue) value changes on high and poor quality connections. On first graph it can be seen bandwidth increases, since excellent throughput is available, until it reaches its maximum threshold value, however simulated increase in RTT value (sudden spike) made no influence in bandwidth value – spike duration is too low for adaptive algorithm to trigger. Lower graph shows how situation look in worst case scenario – sudden spikes of RTT makes no influence at all, bandwidth keeps increasing (video stream starts to lag), sudden increase of lost packets triggers decreasing of bandwidth value, once it is back to normal, algorithm tries to increase bandwidth to stabilize the encoder, but it has already timed out because it took too much time for algorithm to adapt to the network changes.

This experiment shows single capture of network connection inconsistency, while such instability in mobile network happens constantly leading passive adaptive algorithm to lag a lot in respect to actual network state. In cases of wide range fluctuation in network parameters values, lag value contains tenth order of seconds.

4.3 Section generalization

Analysis of WebRTC parameters was performed on static configuration and during active stream between moving vehicle. Data captured during moving vehicle stream brought clear view of how system adjusts itself during constantly changing network environment. Web browsers statistical data allowed to understand problems introduced in adaptive mechanisms, native WebRTC sources explains disadvantages of such mechanism, where adaptation performance is limited to quality of received statistical data. Because no self-decision mechanism is applied in adaptive system, it makes system unable to handle extreme changes in network environment, limiting system to be used on static configuration.

5 DEVELOPMENT AND ANALYSIS OF ADAPTIVE VIDEO STREAM CONTROL SYSTEM

This section presents concept implementation of adaptive video stream control system with intelligent system – neural network predictor, adjusting encoder parameters based on statistical network data. Mechanism is about to be integrated into open source WebRTC real time media stream system environment to perform optimized overall theoretical and practical performance. Algorithm implemented are about to not interfere with current system design.

5.1 Experimental analysis plan

1. Acquisition of statistical network parameters of live media stream.
2. Selection of prediction method based on most accurate prediction with real statistical network parameters.
3. Test of predictor results with stock estimation (reference) parameters.
4. Test of predictor results with two network parameters as input.
5. Compare stock system estimated bandwidth to predicted bandwidth.
6. Propose algorithm to be implemented into WebRTC framework.

5.2 Selection of most suitable prediction method

During acquisition of statistical network parameters of live stream session, with remote endpoint moving in car, causing unstable network to increase latency. Tests performed on 1280x720 (720p) resolution. To select most suitable predictor from reviewed predictors (section 3), predictors are tested with real statistical data captured in section 4. Test data is prepared in such manner to predict arrived frame count based on packets received.

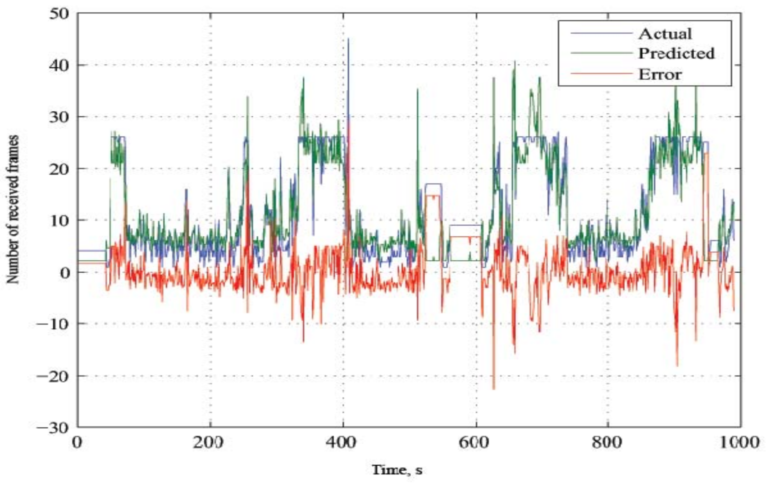


Figure 5.1. Result of trained LN network

Figure 5.1 shows comparison of received frames count versus actual received frames count. Because of linear neuron has linear activation function, neural network response does not react to sudden changes of input parameter (packets received), sometimes produces inadequate change in bitrate predicted, mostly works in similar overall performance to stock WebRTC linear adaptation [27] system, not allowing to adjust encoder parameters correctly. To achieve better results nonlinear activation function with different neural network model should be used.

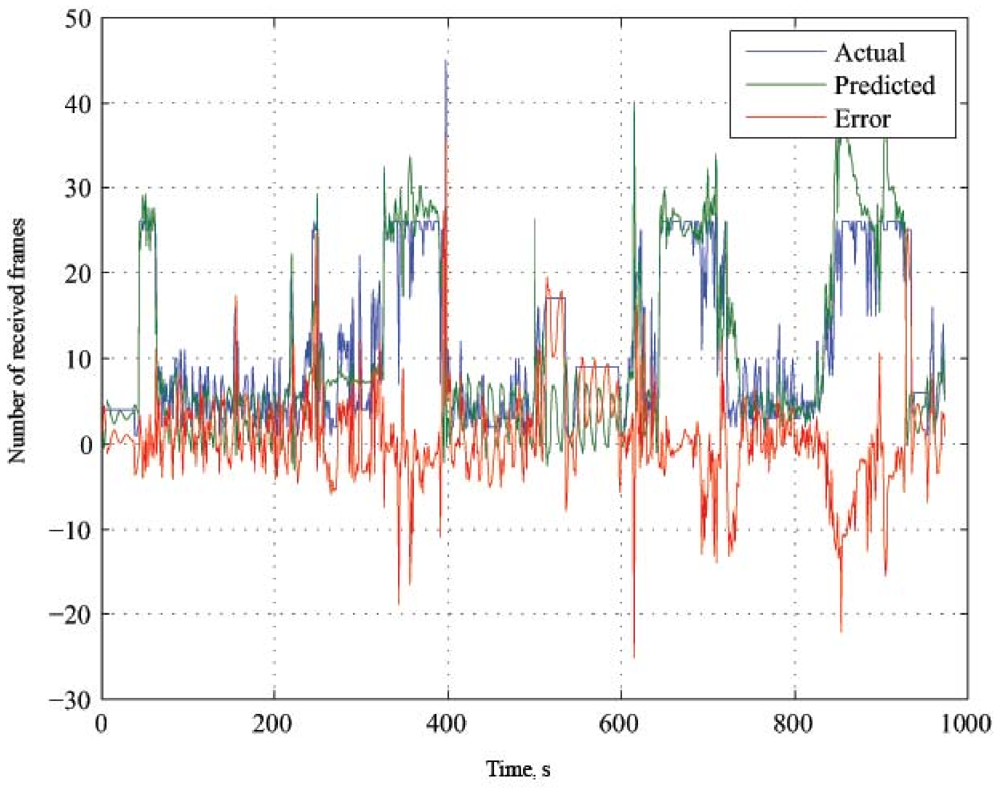


Figure 5.2. Result of trained LRNN network

Figure 5.2 represents same data trained over LRNN network. Predictor is organized with nonlinear activation function having hidden layers, with linear activation function on output neuron. Such network has infinite dynamic response to time series data. Predicted bitrate fluctuates around actual value with high dispersion, it may cause instability if encoder if such parameter will be transferred to encode video stream. Fluctuations happen because neuron model use feedback from hidden layers of predictor, rather than actual output. Complexity of network and arithmetic steps counts are high, taking into account prediction should be calculated between packets receive time interval, output produced should be as close as possible to actual value. Overall performance is much better than linear neuron, it still requires some tuning or external algorithm applied to filter high fluctuations and overshoots in prediction.

To compensate disadvantages of LRNN network, other network model was analyzed – NARX. Main difference between LRNN network is that NARX model use output of network feedback to compensate for errors between actual and previous values, this feature features NARX model as most suitable for prediction of bitrate based on network parameters. Spontaneous spikes are nearly impossible in NARX network, because current predicted value is based on current input and previous output of predictor as well. Figure 5.4 represents predicted value to actual value nearly echoing each other, with negligible overshoots, error values are lowest and at least scattered over whole range in comparison to LRNN and LN models. Model realization requires more storage resources to keep buffers of input and output delays, where in other reviewed models buffers acts as a natural buffer for further arithmetic calculations.

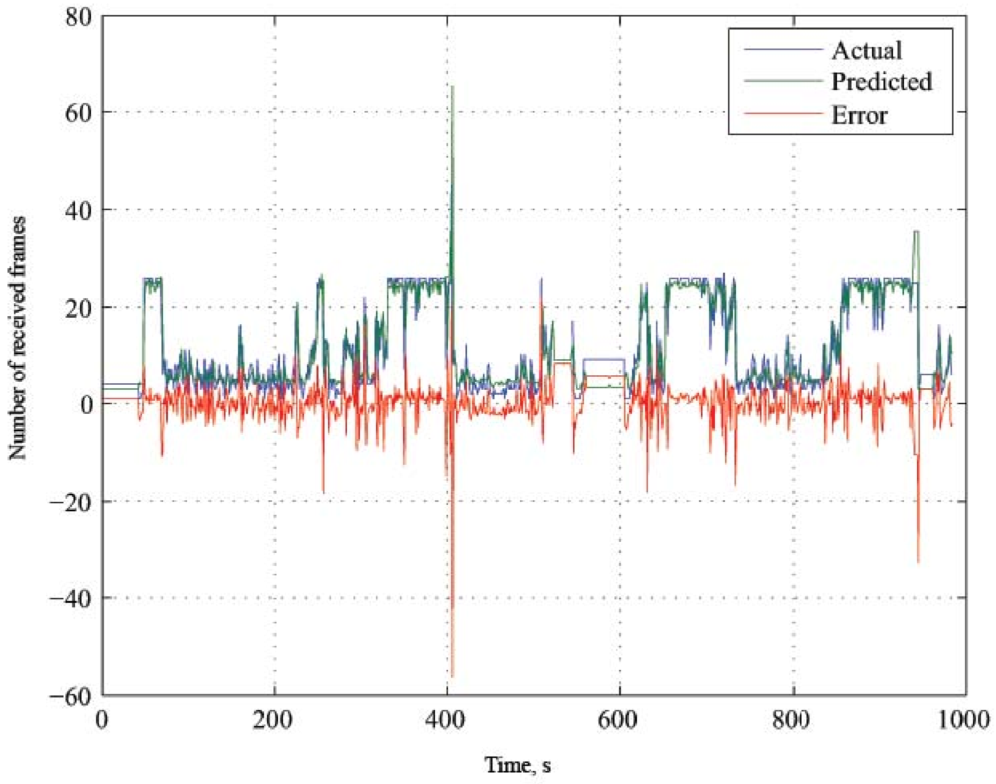


Figure 5.3. Result of trained NARX network

To have more generalized view of neural network models performance, result tables with different network parameters were generated. Table 5.1 shows how each network model performance index varies over different input delays. Best performance is got on NARX model with 2 input delays.

Table 5.1. Predictor performance versus number of delays

Predictor		Input delays					
		2	4	6	8	10	12
NARX	MSE	11.5	12.4	13.5	13.3	13.0	14.6
	STD	3.3	3.5	3.6	3.6	3.5	3.8
LRNN	MSE	16.6	16.2	22.0	20.6	24.5	22.3
	STD	3.9	3.9	4.2	4.4	4.8	4.6

LN	MSE	24.6	24.6	24.6	24.6	24.6	24.6
	STD	4.9	4.9	4.9	4.9	4.9	4.9

Table 5.2 represents variation of performance index over neurons count in hidden layer, resulting in NARX having best performance index over neurons count of 6-24. In LRNN network model error quantity raises a lot if neuron count is increased – parasitic harmonics starts to appear around predicted value. NARX model has little increase in error when neuron count is increased, however accuracy increases noticeably, as well as resources needed for computation of output.

Table 5.2. Predictor performance versus number of neurons

Predictor		Neurons					
		6	8	10	12	16	24
NARX	MSE	11.5	12.5	13.0	12.4	13.2	13.8
	STD	3.3	3.5	3.6	3.5	3.6	3.7
LRNN	MSE	16.2	17.1	16.8	18.8	16.6	25.0
	STD	3.9	4.0	4.0	4.3	3.9	4.7
LN	MSE	24.6	24.6	24.6	24.6	24.6	24.6
	STD	4.9	4.9	4.9	4.9	4.9	4.9

Training samples count comparison is made, results presented in Table 5.3, it is known, the more training samples are applied – the better resulting accuracy is achieved. NARX model shows pretty good results at training samples as low as 50 in count, worst result, as predicted, is achieved on LN network model, which is worse more than in half comparing to NARX model even on 350 samples. Training on Levenberg-Marquardt backpropagation and Bayesian regulation brings similar results, depending on input variable and model structure semantics. However, for NARX model Bayesian regulation brings slightly better result, and best result in overall – 11.5 MSE value.

Table 5.3. Predictor performance versus training samples count

Predictor		Number of training samples				Train algorithm	
		30	50	100	350	LM	BR
NARX	MSE	22.7	15.6	14.3	11.5	13.0	11.5
	STD	4.2	3.8	3.7	3.3	3.5	3.3
LRNN	MSE	25.8	19.9	16.2	16.6	16.2	19.9
	STD	4.2	4.3	3.9	3.9	3.9	4.3
LN	MSE	78.3	29.6	34.4	24.6	24.6	-
	STD	5.4	5.2	5.8	4.9	4.9	-

According to experimental results, NARX predictor shows best overall result based on Bayesian regulation training with 350 training samples using 6 hidden layer neurons and 2 sample delay of external input: mean square error 11.5 with standard deviation in value of 3.3. To perform exact test based on real system parameters, data screened on section 4 was applied. Two most critical parameters describing network state was selected as external inputs of NARX network: RTT value and loss of packets. To perform training, value of output bitrate was manually raised to achieve correct predictions after network is trained.

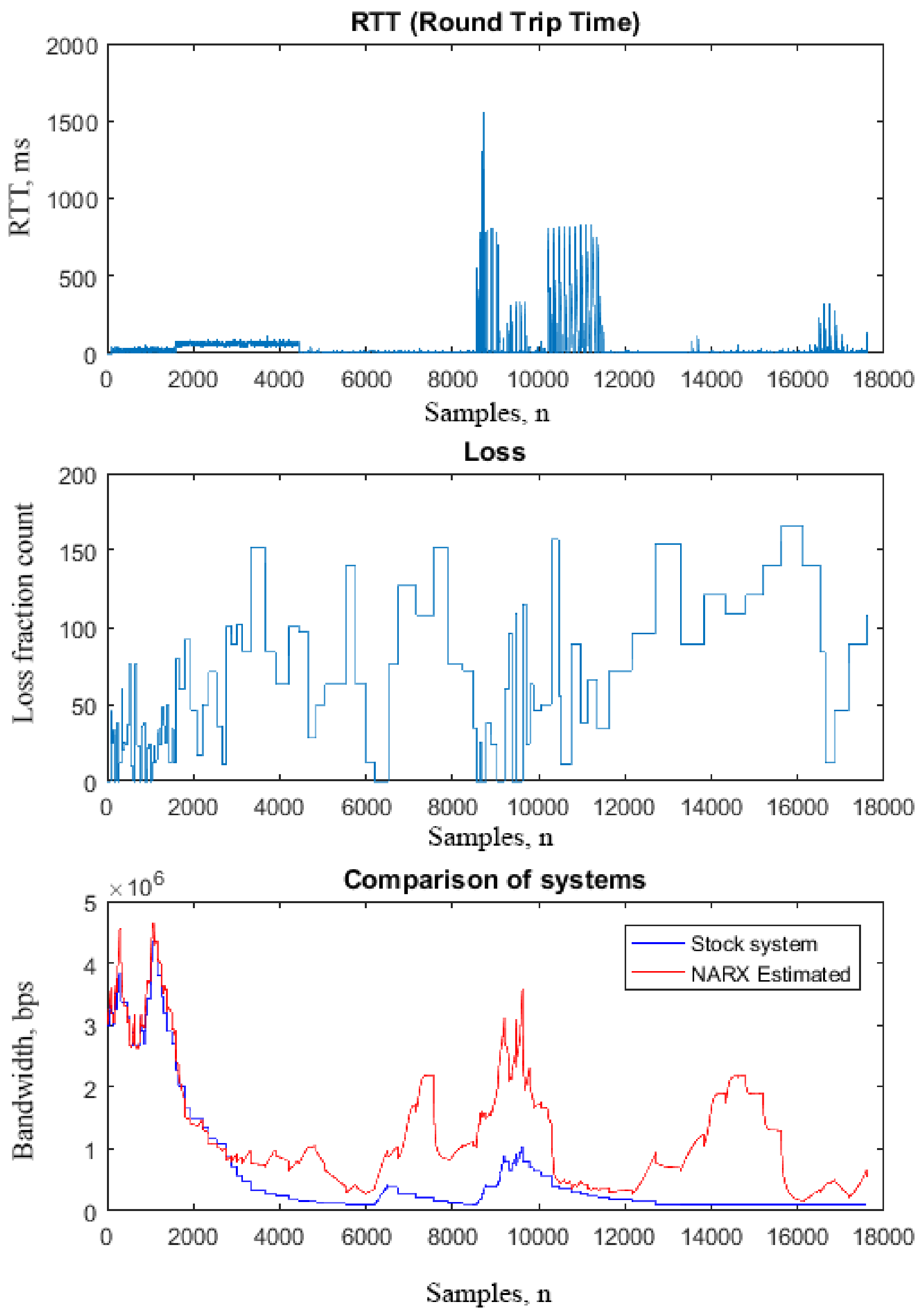


Figure 5.4. Comparison of stock estimation system and NARX predictor

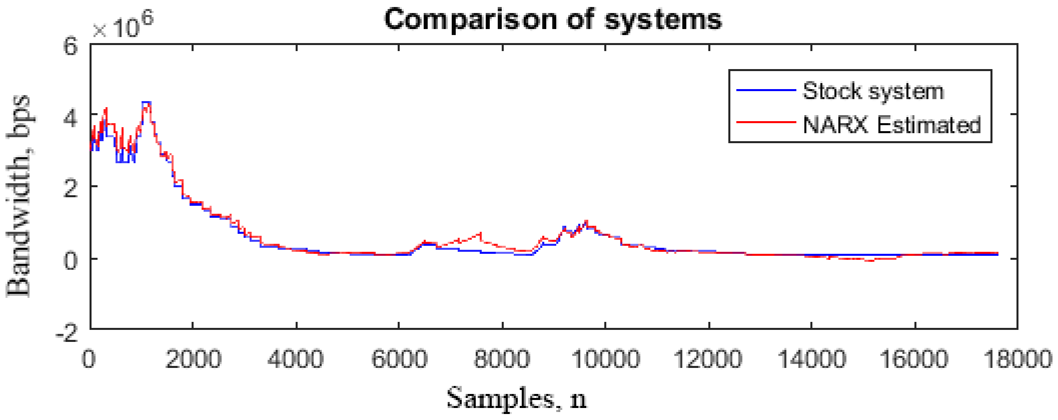


Figure 5.5. Bandwidth estimated by stock system and NARX predictor

Figure 5.5 demonstrates trained NARX network to predict reference bandwidth, estimated by stock system. As inputs, network parameters of round trip time and packet loss are taken. However, such predicted bandwidth value is non well composited, causes encoder to fail to transfer video stream smoothly, retraining with adjusted values was performed. Figure 5.4 represents real prediction results with real statistical data captured during media stream. Network connection is stable, however, as remote endpoint is a mobile device in motion, causing RTT value to fluctuate, some loss is introduced. Stock system estimated bitrate (blue) reacts to network changes, loss is detected, bitrate starts to decrease, then conditions normalizes, and bitrate start to rise. At the moment bitrate should rise, RTT delays are introduced, estimation system starts to lag, leading to flat estimated bitrate and complete loss of stream connection. NARX predictor (red line) reacts to changes immediately handling encoder to provide smooth image quality increasing macroblocks, after extreme conditions gone, bitrate prediction mechanism recovers and start to rise, macroblock size is decreased, high quality image is transferred. Ability to react to both variables (RTT, loss) independently is great feature of NARX network, providing best overall result for WebRTC technology in condition when mobile device with mobile network is active. Intelligent adaptive algorithms in decision critical environments is more efficient than using static rule based techniques, which tend to go to abnormal results if main dependent variables suddenly changes in value or goes outside described ranges.

5.3 Implementation into WebRTC framework

To implement intelligent adaptive into open source WebRTC framework, two most suitable ways are found: using open source NARX C++ library; expressing mathematical functions directly. Applying of dedicated library is the easiest way, but it has some drawbacks: incorporation of third party libraries to WebRTC framework is rather complicated, effectiveness of library may be worse than pure mathematical model implementation, some limitations, like input values ranges may be inconsistent. Pure mathematical representation may be optimized well, direct structure is accessible, trained network weight coefficients may be applied directly from other environment.

Mathematical model for transferring predictor from "Matlab" programming interface is provided in this section. Figure 5.6 represents "Matlab" software generated NARX network model with two input variables (RTT, loss), two input delays are introduced, 10 hidden layers provided with nonlinear activation functions.

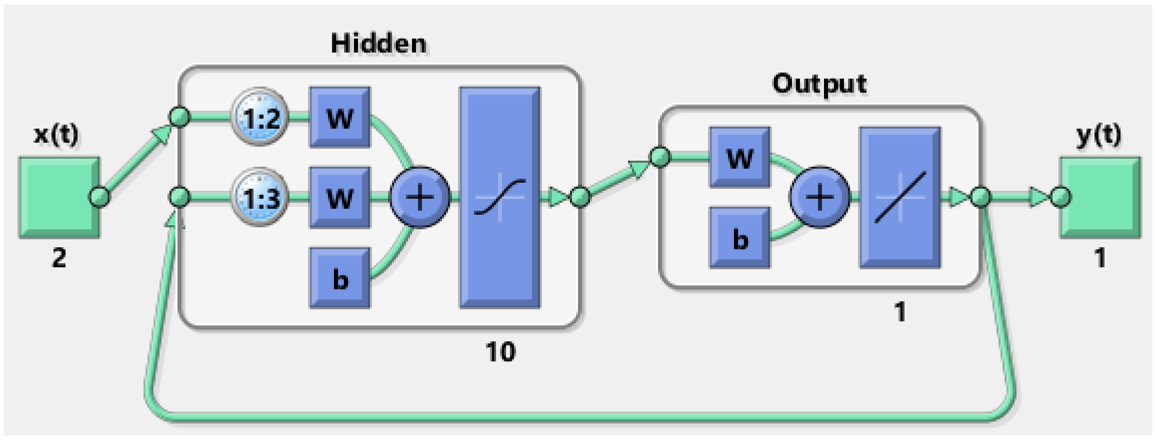


Figure 5.6. "Matlab" software generated NARX model

To perform training, Bayesian regulation algorithm was selected since it shows slightly better results, iterations count limited to 5000 iterations, mean squared error set to threshold of 1^{-6}. Training operation succeed in time of 8 minutes and 27 seconds with 4625 iterations, MSE value 9.89^{-7}.

To apply NARX network in C++ environment mathematical representation is denoted. Inputs denoted as $x(n)$ and $z(n)$ accordingly, output of predictor $-y(n)$. Mathematical representation follows:

$$\begin{aligned} v(n) = x(n)w_1 + x(n-1)w_2 + x(n-2)w_3 + y(n-1)w_6 + y(n-2)w_5 + y(n-3)w_4 + \\ + w_0 + z(n)w_7 + z(n-1)w_8 + z(n-1)w_9, \end{aligned} \quad \textbf{(5.1)}$$

where $v(n)$ – output before activation function, w_0 – neuron bias coefficient, $w_{1..9}$ – weight coefficients for corresponding input or feedback value.

To calculate output of predictor, nonlinear activation function applied:

$$y(n) = \frac{1}{1+e^{-v(n)}}; \quad \textbf{(5.2)}$$

If additional training steps during prediction are necessary, training can be accomplished by calculating error value between predicted and actual value:

$$err(n) = y_{\text{actual}}(n) - y(n), \quad \textbf{(5.3)}$$

updating coefficients accomplished by recalculation according to error:

$$\begin{aligned} & w_1(n+1) = w_1(n) + \mu \times err(n) \times x(n); \\ & w_2(n+1) = w_2(n) + \mu \times err(n) \times x(n-1); \\ & \ldots \\ & w_0(n+1) = w_0(n) + \mu \times err(n), \end{aligned} \quad \textbf{(5.4)}$$

where μ – is value in range of 0..1. Training calculations are performed until sum product of error is below some threshold value, or iterations count of training reaches its set limit.

Mathematical implementation of NARX model in C++ environment is easy, and requires not so many resources. One of most resources requiring step is calculation of exponential value, but as modern computers and mobile devices central processing units has intended hardware modules to process such problems, there is no problem about it.

To perform tests for remote endpoint HTML and JavaScript programming languages was applied, web page application acting as an endpoint was developed. Application displays incoming and outgoing stream, allows to select resolution of video stream captured by hardware video device, allows to set conference member name. Figure 5.7 illustrates graphical user interface of web application. Main goal of web based application is to allow to use WebRTC on various devices, where implementation of dedicated application is too complex and is non mandatory. For this case we application acts as a bridge between different hardware architectures sharing equal capability – capability of modern web browsers and power of HTML5 standard incorporated into them.

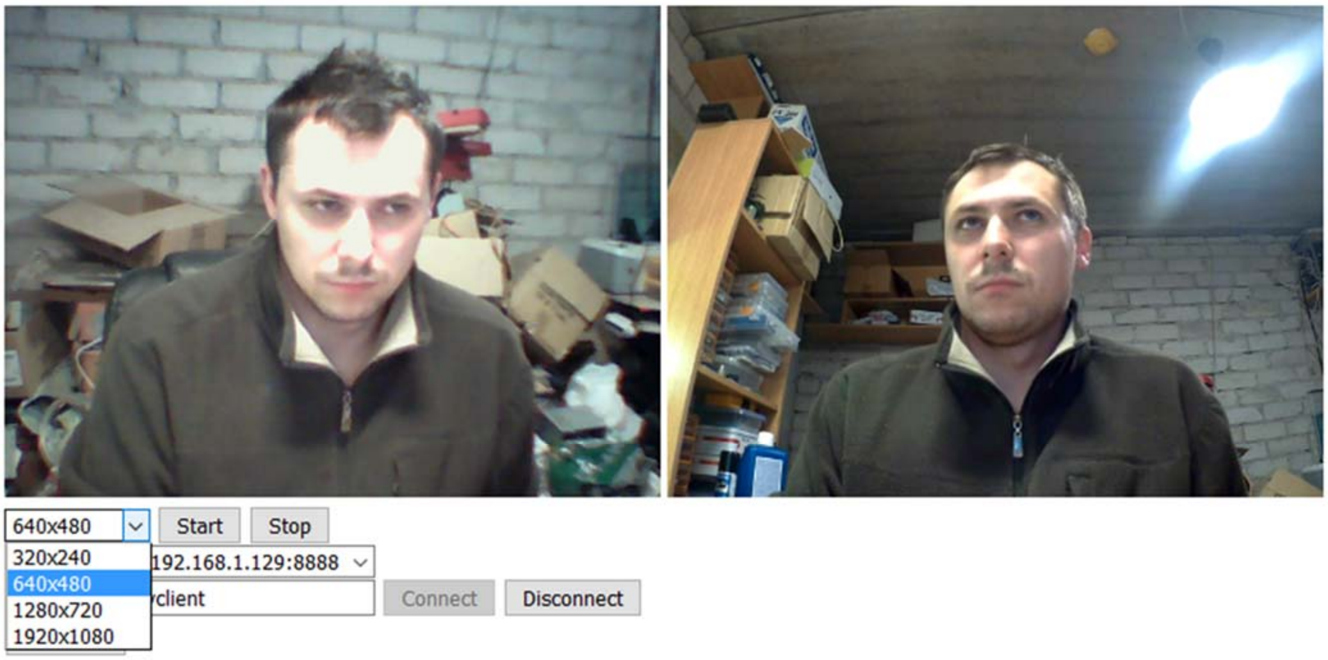


Figure 5.7. WebRTC based web application

Figure 5.8 illustrates real time video conference graphical interface in WebRTC native application. Behind graphical interface there is Google Chrome web browser environment, which emulates environment of web browser enabling HTML5 technology to be used as standalone application.

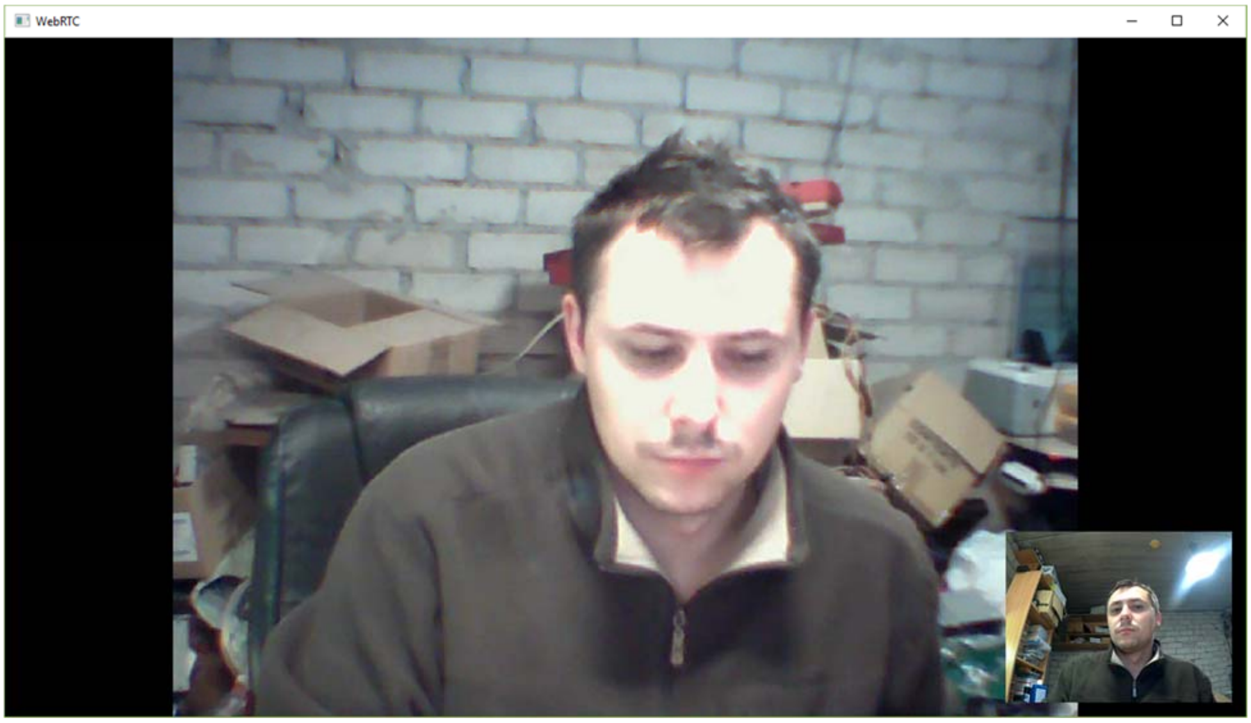


Figure 5.8. WebRTC based native application

During the research timeframe a lot of fixes and new features were developed for WebRTC framework. One of the most interesting feature is ability to stream single media stream to several remote endpoints (currently maximum of 4 at once). Since peer-to-peer technology allows direct connection only between two endpoints, multiplexing technique is applied (Figure 5.9). Such scheme is implemented through local buffer doubling, where separate emulated browser environments are created, each performing own statistics on network quality.

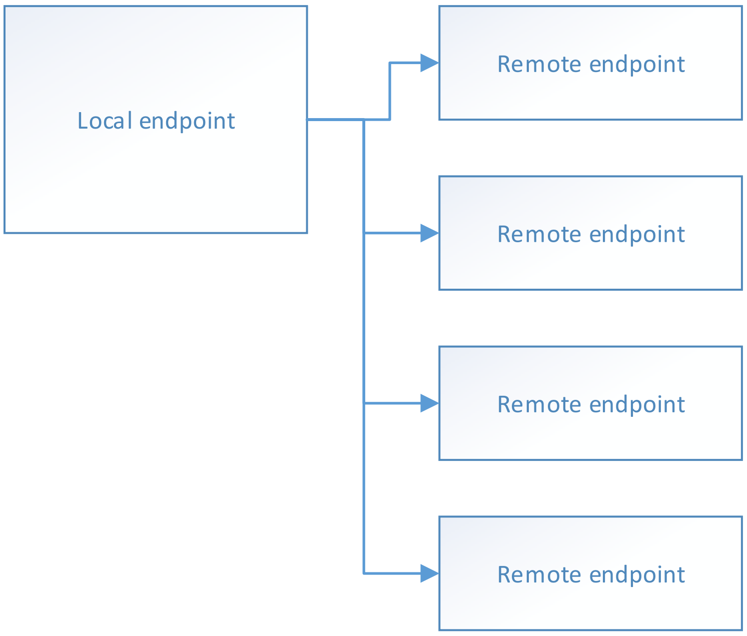


Figure 5.9. One-to-many multiplexing

Multi-connection interface allows of controlling each codec interface separately, meaning intelligent bitrate predictors could be developed in such manner where each adaptively adjust each connection parameters, or parameters could be globalized making adaptive system more robust where all parameters are summed, predicted, and equal parameters passed to all codec. Such scheme will discriminate endpoints with perfect connection quality lowering overall media stream quality. During experiments, because of low network throughput and devices hardware resources, maximum remote endpoints count was limited to 3. Further development of multi-stream adaptive predictor is open.

5.4 Section generalization

Prediction methods and its parameters were analyzed, mean square error and standard deviation parameters were compared between each other with equivalent parameters set, NARX predictor were selected as best method to predict bitrate in WebRTC technology. Mathematical representation of model with trained parameters is provided, allowing algorithm to be implemented practically in any programming environment. Practical comparison demonstrates features where intelligent adaptive predictor predicts bitrate parameter more efficient than stock system, thus optimizing overall system stability when used on mobile network with jittery mobile connection. Optimizations allows encoder to be pushed to its theoretical limits, ensures maximum usage of network bandwidth available.

6 CONCLUSIONS

Aim of the work is achieved by performing analysis of prediction methods and their applicability to predict data bandwidth of the real time video streaming. After performing analytical review of adaptive video streaming systems, analysis of statistical data screening, and development of adaptive video stream control system, conclusions were formulated:

1. After performing analytical review, it can be said, that:
   a) Demand on adaptive real time streaming systems is high, special attention is paid to external plugin free systems because of its universality and applicability on most modern devices.
   b) WebRTC is unambiguous leader in technologies of new generation real time media stream systems, it is proven by fact WebRTC is used as parent technology on developing best known real time video conference and presentation products.
   c) Publicly available, open source code of WebRTC technology is optimized and close sourced to maintain competition in real time media streaming niche while maintaining potential customers by presenting unique features in the market.
   d) Well performing real time media stream technology is about to step in and replace current standards in areas like IP television, IP telephony, social media.
2. Analysis of prediction methods gives knowledge to conclude, that:
   a) Artificial neural network algorithms perform better than traditional rule based algorithms providing much more accurate results of prediction, while despite high complexity still being implementable on most devices.
   b) Correct selection of training technique and training datasets plays critical role in terms of predictor accuracy and overall performance.
   c) Externally trained network, which does not require constant retraining, can be easily parsed to mathematical representation of simple arithmetic operations.
   d) Neural networks with feedback and samples delays are well suitable long term signals because of infinite dynamic response on time series.
   e) Levenberg-Marquardt backpropagation or Bayesian regulation optimization functions decrease error of neural network training session significantly.
3. Investigation on statistical data collected during real time media stream results in:
   a) WebRTC technology has poor adaptive encoder parameters predictor applied which leads lag of video stream or complete loss of connection.
   b) WebRTC adaptive mechanism relays on remote endpoint reported maximum bitrate available.

   c) Most important accessible statistical parameters for prediction of bitrate available are: round trip time; packet loss fraction.
   d) WebRTC technology additionally adjusts encoder parameters in case of CPU overuse or high quantization parameter value is detected.
   e) Current non intelligent adaptive mechanism does not react to sudden control parameters changes, does have strict thresholds in adaptation range, limiting high throughput networks to reach maximum quality.
4. Development and analysis of adaptive video stream control system, it can be concluded, that:
   a) Statistical data collected during real time media stream session is well suitable to predict bitrate of encoder.
   b) NARX neural network provide good performance in terms of prediction accuracy and error rate with leaving MSE at value of 11.5, meaning frames could be predicted with an average error of 3-4 frames.
   c) Best performance of NARX neural network is achieved by training network with 350 samples, while applying Bayesian regulation function and configuring network to have 2 input samples delay with 6 neurons in hidden layer.
   d) Mathematical representation of NARX network is easily implementable in WebRTC framework without inferring with actual system design, leaving calculation of nonlinear activation function exponential value the most demanding arithmetical operation.
5. Latest update on WebRTC technology features ability for multiple end nodes to be connected to single stream performing multiplexing of media stream channel, which provide ability for future work, optimizations on developing multi-channel predictors for each peer connection to have separate adaptive control.

References

1. S. J. Vaughan-Nichols, “Flash is dead. Long live HTML5.,” *ZDNet*. 2011.
2. A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, “Will IPTV ride the peer-to-peer stream? [Peer-to-peer multimedia streaming],” *IEEE Commun. Mag.*, vol. 45, no. 6, pp. 86–92, 2007.
3. S. Dutton, *Getting Started with WebRTC*. 2014.
4. C. Perkins and M. Westerlund, “Multiplexing RTP Data and Control Packets on a Single Port,” *Internet Soc. RFC 5761*, pp. 1–13, 2010.
5. R. Jesup, S. Loreto, Ericsson, and M. Tuexen, “WebRTC Data Channels,” *Netw. Work. Gr.*, pp. 1–16, 2015.
6. N. Brice, J. Tanke, D. Frey, P. Molli, A. Mostefaoui, N. Brice, J. Tanke, D. Frey, P. Molli, A. Mostefaoui, B. Nédelec, and D. Frey, “SPRAY : an Adaptive Random Peer Sampling Protocol,” 2015.
7. F. R. N. Barbosa and L. F. G. Soares, “Towards the Application of WebRTC Peer-to-Peer to Scale Live Video Streaming over the Internet,” *Simpósio Bras. Redes Comput.*, no. Figure 1, pp. 119–124, 2014. < http://sbrc2014.ufsc.br/anais/files/wp2p/ST4-1.pdf, checked: 2016-05-31>
8. L. L. Fernandez, M. P. Diaz, R. B. Mejias, F. J. Lopez, and J. A. Santos, “Kurento: A media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC,” in *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013*, 2013.
9. K. Schiller, “TokBox: Video Chat Without the Hassle,” *Inf. Today*, vol. 28, no. 1, p. 36, 2011.
10. J. D. Cole, “OpenTok $^{\text{TM}}$: A Free Open Source API for Video Conferencing in Distance Education Department of Instructional Technology,” no. 2, pp. 1–7.
11. S. A. Baset and H. G. Schulzrinne, “An analysis of the Skype peer-to-peer internet telephony protocol,” in *Proceedings - IEEE INFOCOM*, 2006.
12. A. Asiri and L. Sun, “Performance Analysis of Video Calls Using Skype,” pp. 155–162.
13. J. Bankoski, P. Wilkins, and Y. Xu, “Technical overview of VP8, an open source video codec for the web,” in *Proceedings - IEEE International Conference on Multimedia and Expo*, 2011.
14. P. C. Bryan and K. Zyp, “JSON Reference,” *Internet Engineering Task Force (IETF) Draft*, 2012. [Online]. Available: <http://tools.ietf.org/html/draft-pbryan-zyp-json-ref-03, checked 2016-05-31>.
15. A. B. Roach, “WebRTC Video Processing and Codec Requirements,” *Netw. Work. Gr.*, pp. 1–9, 2015.
16. C. Kilinc and K. Andersson, “A congestion avoidance mechanism for WebRTC interactive video sessions in LTE networks,” *Wirel. Pers. Commun.*, vol. 77, no. 4, pp. 2417–2443, 2014.
17. P. Tumas and A. Serackis, “Peer-to-peer adaptive video streaming system”, *Electronic and Electrical Engineering (AIEEE), 2015 IEEE 3rd Workshop on Advances in 13-14 Nov. 2015*
18. J. Skirelis and A. Serackis, 2015. “Prediction of the real-time video streaming performance based on the peer connection statistics” *Electronic and Electrical Engineering (AIEEE), 2015 IEEE 3rd Workshop on Advances in 13-14 Nov. 2015*

19. V. Jasevičiūtė, D. Plonis, A. Serackis, "Dynamic adaptation of the jitter buffer for video streaming applications", *IEEE 2nd Workshop on advances in information, electronic and electrical engineering (AIEEE), proceedings of the 2nd workshop November 28–29, 2014 Vilnius, Lithuania.*
20. Š. Paulikas, Estimation of video quality of H.264/AVC video streaming", *EUROCON 2013: Zagreb, Croatia*.
21. Š.Paulikas, P. Sargautis and V. Banevičius, "Impact of wireless channel parameters on quality of video streaming". *Elektronika ir elektrotechnika. ISSN 1392-1215. 2011, No. 2 (108), p. 27-30*
22. A. Khan, L. Sun, E. Ifeachor, J. O. Fajardo, and F. Liberal, "Video quality prediction model for h.264 video over umts networks and their application in mobile video streaming," *Communications (ICC)*, 2010 IEEE International Conference on, May 2010, pp. 1–5
23. M. Chen, "Amvsc: A framework of adaptive mobile video streaming in the cloud," *Global Communications Conference (GLOBECOM)*, 2012 IEEE, Dec 2012, pp. 2042–2047.
24. L. Anegekuh, L. Sun, and E. Ifeachor, "End to end video quality prediction for hevc video streaming over packet networks," *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on. IEEE*, 2013, pp. 1–6.
25. B. Steyaert, K. Laevens, D. De Vleeschauwer, and H. Bruneel, "Analysis and design of a playout buffer for VBR streaming video," *Annals of Operations Research, vol. 162, no. 1*, pp. 159–169, 2008.
26. M. Baba, H. Kurokawa, and Y. Kato, "Buffer-based lowdelay playout control methods for IPTV terminals," *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE. IEEE*, 2009, pp. 1–6.
27. T.-Y. Huang, R. Johari, N.McKeown,M. Trunnell, and M.Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *Proc. ACM SIG- COMM, 2014*, pp. 1–14.
28. D. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *Signal Processing, IEEE Transactions on, vol. 39, no. 1*, pp. 92–114, 1991.
29. D. T. Slock and T. Kailath, "A modular prewindowing framework for covariance ftf rls algorithms," *Signal Processing, vol. 28, no. 1*, pp. 47–61, 1992.
30. ——, "A modular multichannel multiexperiment fast transversal filter rls algorithm," *Signal processing, vol. 28*, no. 1, pp. 25–45, 1992.
31. S. S. Haykin, Adaptive filter theory. Pearson Education India, 2008.
32. S. Kollias and D. Anastassiou, "An adaptive least squares algorithm for the efficient training of artificial neural networks," *Circuits and Systems, IEEE Transactions on, vol. 36, no. 8*, pp. 1092–1101, 1989.
33. D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation, vol. 4, no. 3*, pp. 448–472, 1992.
34. A. Berkvist, D. Burnett, C. Jennings, A. Narayanan, (2011) "WebRTC 1.0: RealTime Communication Between Browsers". Working Draft.
35. F. Bronzino, R. Gaeta, M. Grangetto, G. Pau, G. (2012) "An Adaptive Hybrid CDN/P2P Solution for Content Delivery Networks". *VCIP*, page 1-6, IEEE.

36. S. Cho, J. Cho, S. Shin, (2010) “Playback Latency Reduction for Internet Live Video Services in CDN-P2P Hybrid Architecture”. *2013 IEEE International Conference on Communications.*
37. C. Huang, A. Wang, J. Li, K. Ross, (2008) “Understanding hybrid CDN-P2P: why limelight needs its own Red Swoosh”. *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video.*
38. “Generic Coding of Moving Pictures and Associated Audio Information-Part 2: Video,” *ITU-T and ISO/IEC JTC 1, ITU-T Recommendation H.262 and ISO/IEC 13 818-2 (MPEG-2)*, 1994.
39. “Video Coding for Low Bit Rate Communication,” *ITU-T, ITU-T Recommendation H.263 version 1*, 1995.
40. “Advanced Video Coding for Generic Audiovisual Services,” *ITU-T, ITU-T Recommendation H.264*, November 2007.
41. J. Saghri, A. Tescher, and J. Reagan, “Practical transform coding of multispectral imagery,” *Signal Processing Magazine, IEEE*, pp. 32–43, 2005.
42. C. Loeffler, A. Ligtenberg, and G. S. Moschytz, “Practical fast 1-D DCT algorithms with 11 multiplications,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '89), vol. 2*, pp. 988–991, Glasgow, UK, May 1989.
43. H.S. Malvar, A. Hallapuro, M. Karczewicz, L. Kerofsky, “Low-complexity Transform and Quantization in H.264/AVC,” *IEEE Transactions on Circuit and Systems for Video Technology, Vol. 13, No. 7*,pp. 598-603, July 2003.
44. J. Bankoski, “On2’s Truemotion VP7 video codec and golden frames”, EE Times, Jul 2008.
45. D. Marpe, T. Wiegand, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard,” *IEEE Transactions on Circuit and Systems for Video Technology, Vol.13, No. 7*, pp. 620-635, July 2003

APPENDICES

UML diagrams of WebRTC signaling server and client software

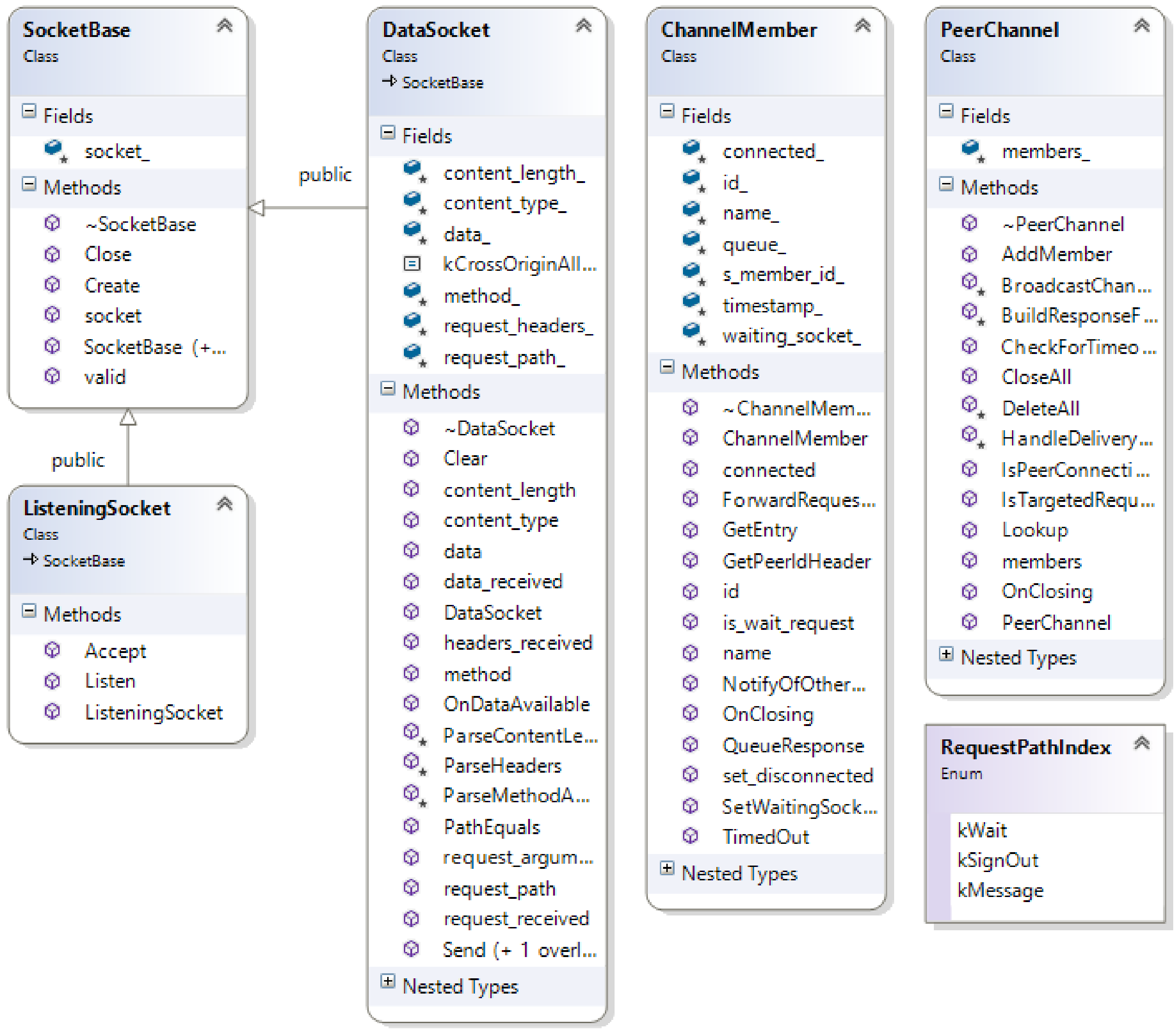


Figure 1. UML diagram of signaling server software

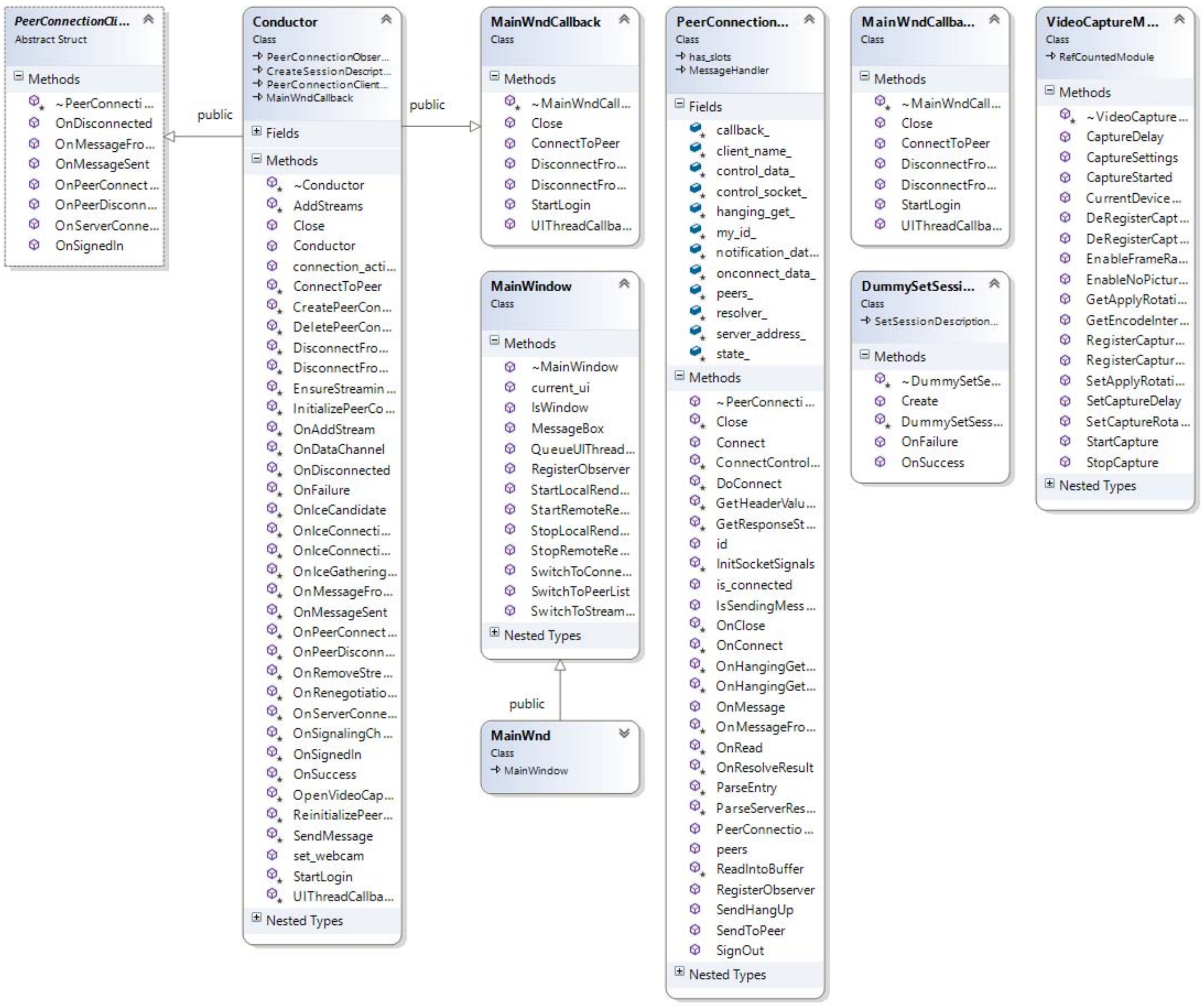


Figure 2. UML diagram of client software

Materials of presentation given at 19th Conference of Lithuanian Junior Researches

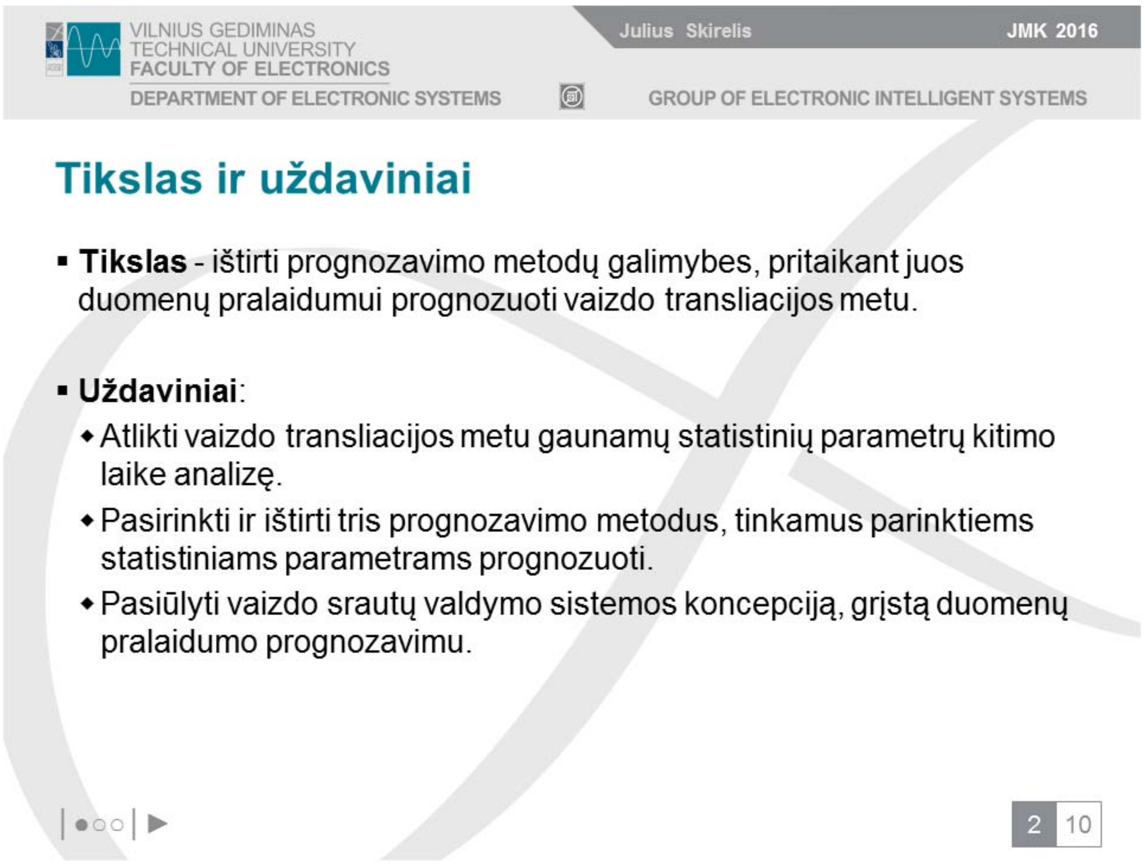
VILNIUS GEDIMINAS
TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS

DEPARTMENT OF ELECTRONIC SYSTEMS
GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Vaizdo srautų valdymo metodų tyrimas

Julius Skirelis

Kovo 18d.
Vilnius



VILNIUS GEDIMINAS
TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

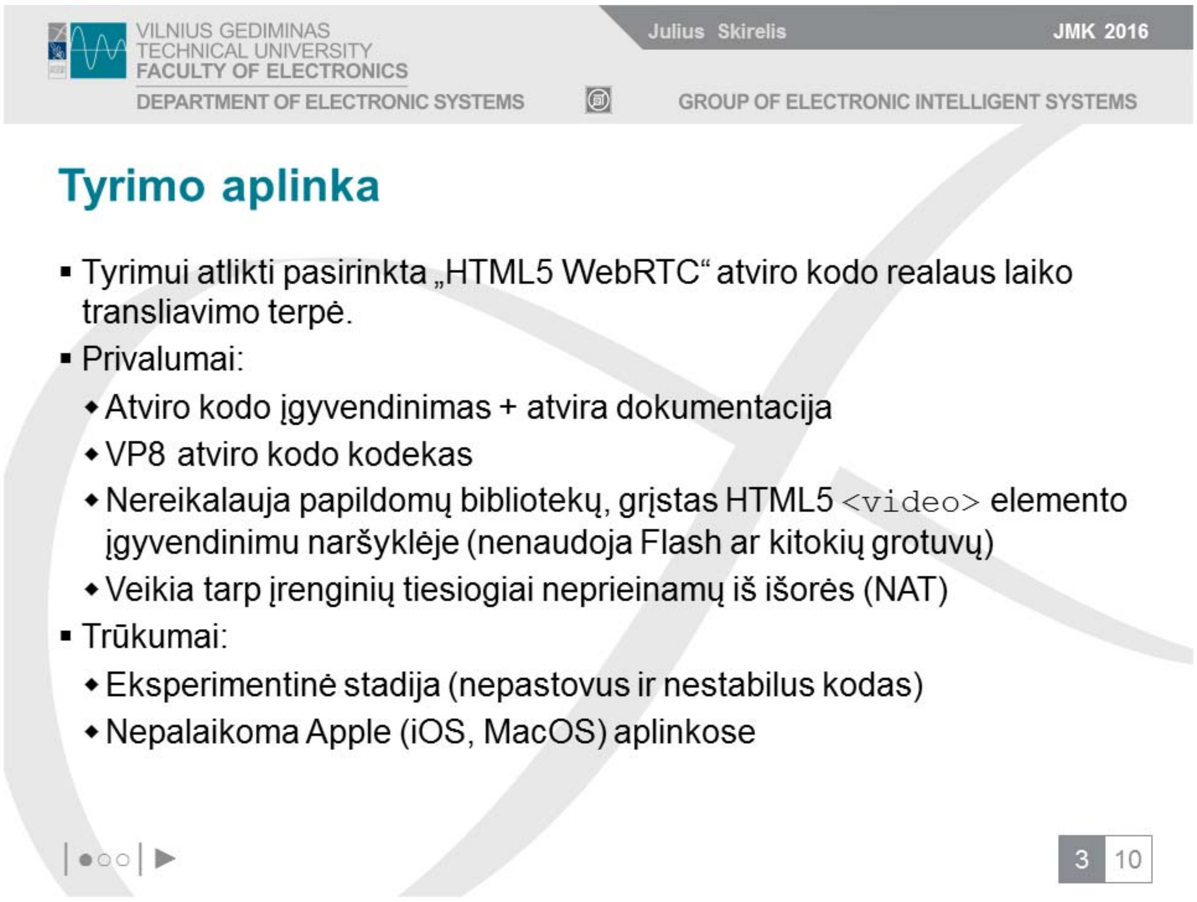
Julius Skirelis JMK 2016

GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Tikslas ir uždaviniai

- **Tikslas** - ištirti prognozavimo metodų galimybes, pritaikant juos duomenų pralaidumui prognozuoti vaizdo transliacijos metu.
- **Uždaviniai**:
  - Atlikti vaizdo transliacijos metu gaunamų statistinių parametrų kitimo laike analizę.
  - Pasirinkti ir ištirti tris prognozavimo metodus, tinkamus parinktiems statistiniams parametrams prognozuoti.
  - Pasiūlyti vaizdo srautų valdymo sistemos koncepciją, grįstą duomenų pralaidumo prognozavimu.

2 10



VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

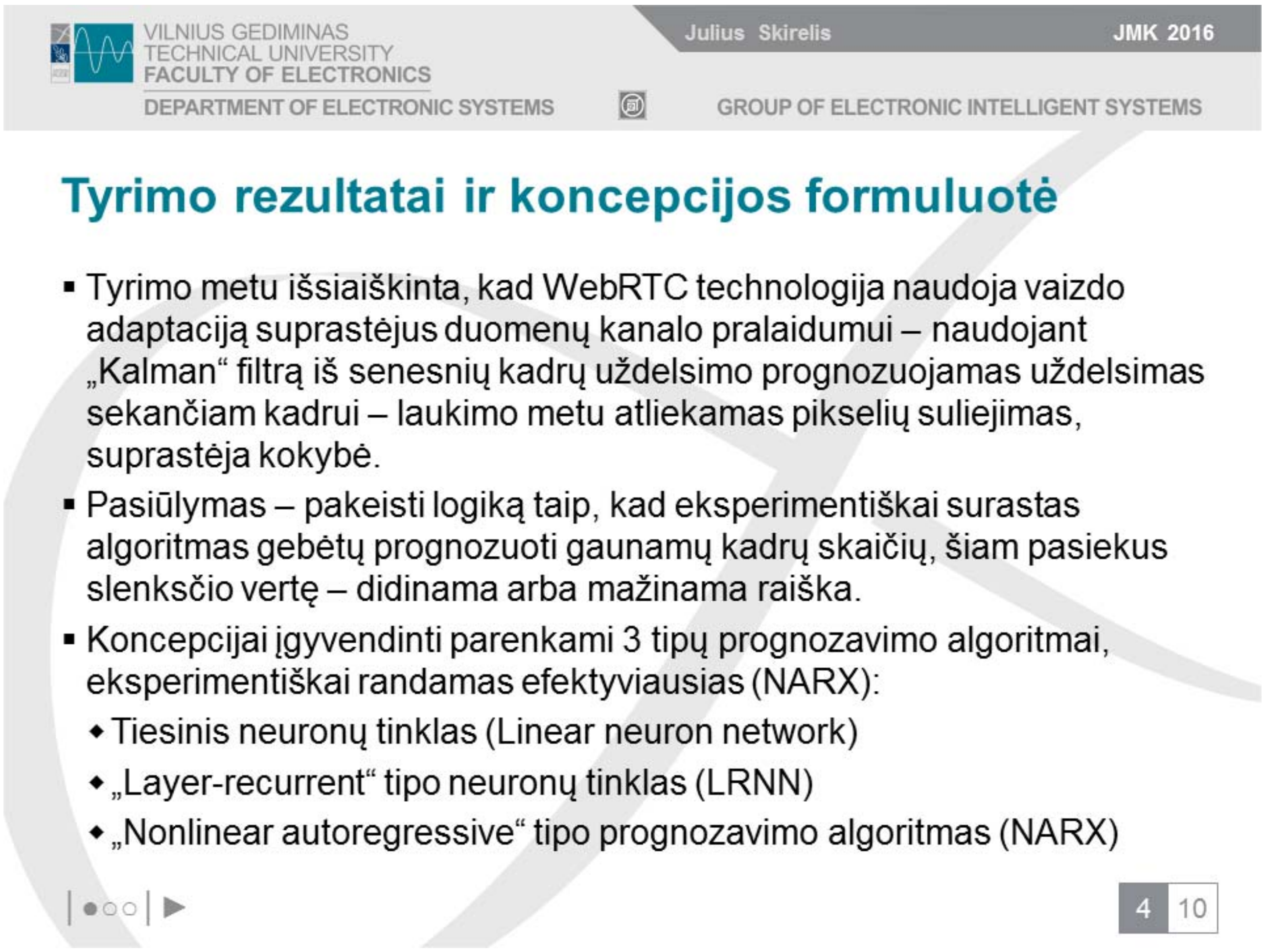
Julius Skirelis JMK 2016

GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Tyrimo aplinka

- Tyrimui atlikti pasirinkta „HTML5 WebRTC“ atviro kodo realaus laiko transliavimo terpė.
- Privalumai:
  - Atviro kodo įgyvendinimas + atvira dokumentacija
  - VP8 atviro kodo kodekas
  - Nereikalauja papildomų bibliotekų, grįstas HTML5 `<video>` elemento įgyvendinimu naršyklėje (nenaudoja Flash ar kitokių grotuvų)
  - Veikia tarp įrenginių tiesiogiai neprieinamų iš išorės (NAT)
- Trūkumai:
  - Eksperimentinė stadija (nepastovus ir nestabilus kodas)
  - Nepalaikoma Apple (iOS, MacOS) aplinkose

3 10



VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

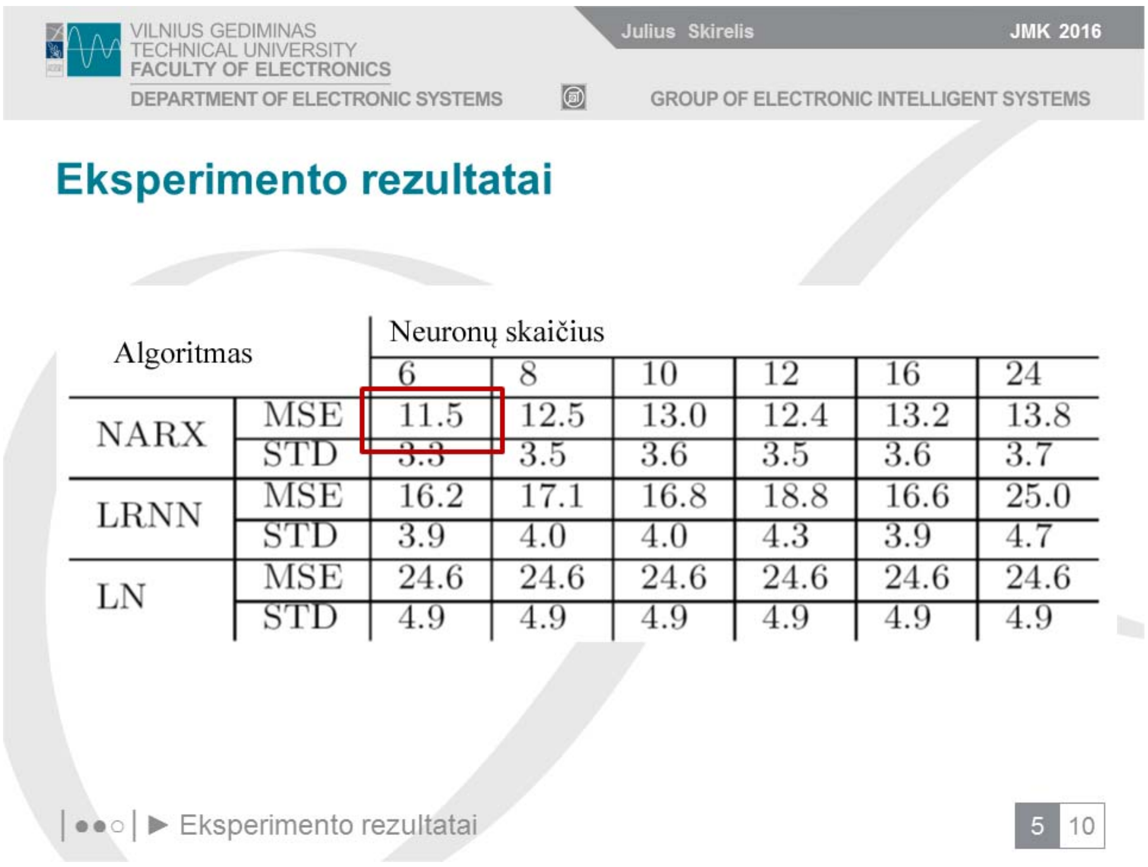
Julius Skirelis JMK 2016

GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Tyrimo rezultatai ir koncepcijos formuluotė

- Tyrimo metu išsiaiškinta, kad WebRTC technologija naudoja vaizdo adaptaciją suprastėjus duomenų kanalo pralaidumui – naudojant „Kalman“ filtrą iš senesnių kadrų uždelsimo prognozuojamas uždelsimas sekančiam kadrui – laukimo metu atliekamas pikselių suliejimas, suprastėja kokybė.
- Pasiūlymas – pakeisti logiką taip, kad eksperimentiškai surastas algoritmas gebėtų prognozuoti gaunamų kadrų skaičių, šiam pasiekus slenksčio vertę – didinama arba mažinama raiška.
- Koncepcijai įgyvendinti parenkami 3 tipų prognozavimo algoritmai, eksperimentiškai randamas efektyviausias (NARX):
  - Tiesinis neuronų tinklas (Linear neuron network)
  - „Layer-recurrent“ tipo neuronų tinklas (LRNN)
  - „Nonlinear autoregressive“ tipo prognozavimo algoritmas (NARX)

4 10



VILNIUS GEDIMINAS TECHNICAL UNIVERSITY FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

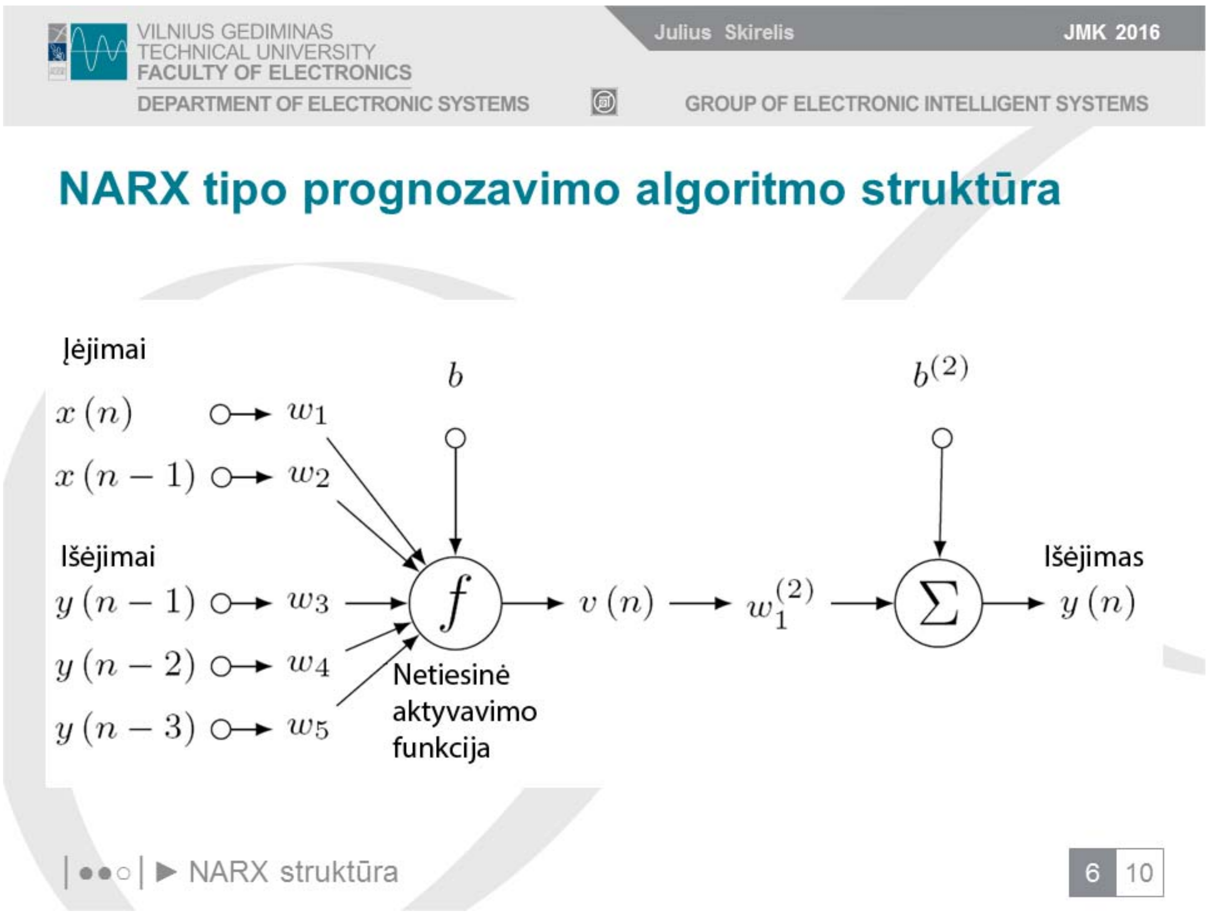
Julius Skirelis JMK 2016
GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Eksperimento rezultatai

Algoritmas		Neuronų skaičius					
		6	8	10	12	16	24
NARX	MSE	11.5	12.5	13.0	12.4	13.2	13.8
	STD	3.3	3.5	3.6	3.5	3.6	3.7
LRNN	MSE	16.2	17.1	16.8	18.8	16.6	25.0
	STD	3.9	4.0	4.0	4.3	3.9	4.7
LN	MSE	24.6	24.6	24.6	24.6	24.6	24.6
	STD	4.9	4.9	4.9	4.9	4.9	4.9

| ●●○ | ► Eksperimento rezultatai

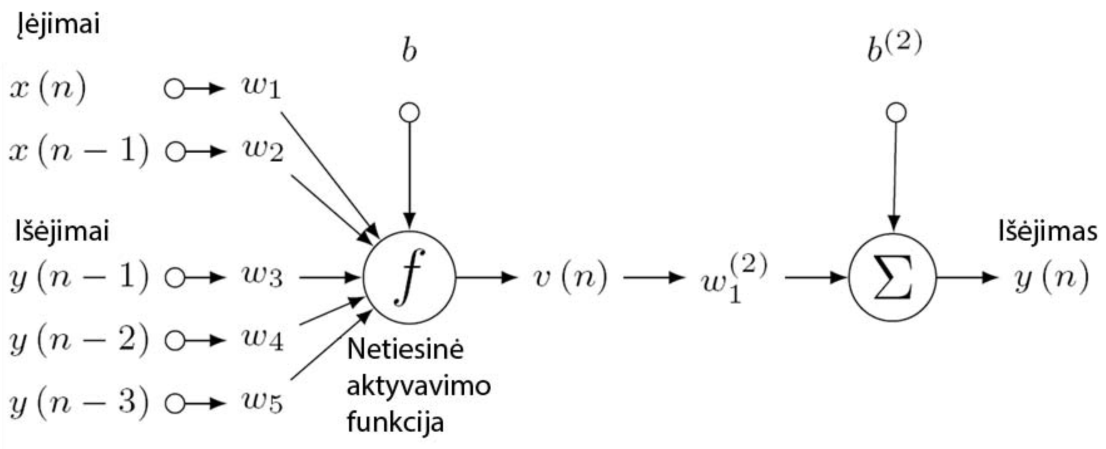
5 10



VILNIUS GEDIMINAS TECHNICAL UNIVERSITY FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Julius Skirelis JMK 2016
GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

NARX tipo prognozavimo algoritmo struktūra



| ●●○ | ► NARX struktūra

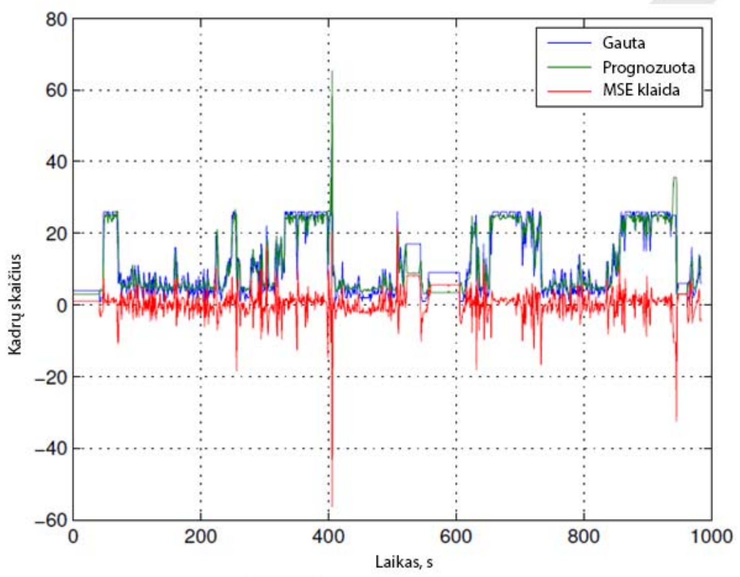
6 10

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Julius Skirelis JMK 2016

GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

NARX prognozavimo algoritmo rezultatai



|●●○| ► NARX rezultatai

7 10

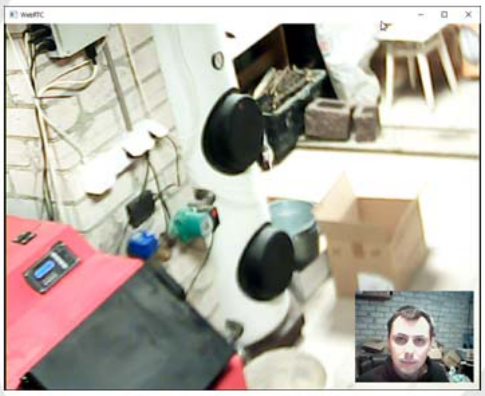
VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

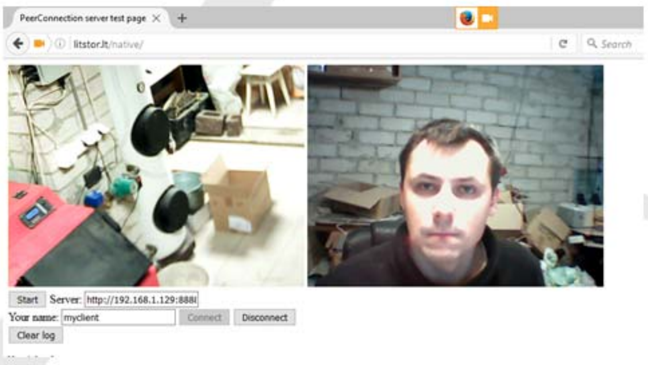
Julius Skirelis JMK 2016

GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Eksperimentinė patikra

- Patikros metu sukuriama vaizdo konferencija tarp įrenginių įterpiant dirbtinius srauto trikdžius (pralaidumo limitas, atsitiktinis paketų praradimas, ryšio nestabilumas), stebimas vaizdas, paveikslo kokybė, palyginami srauto statistiniai parametrai.





PeerConnection server test page

litstor.lt/native/

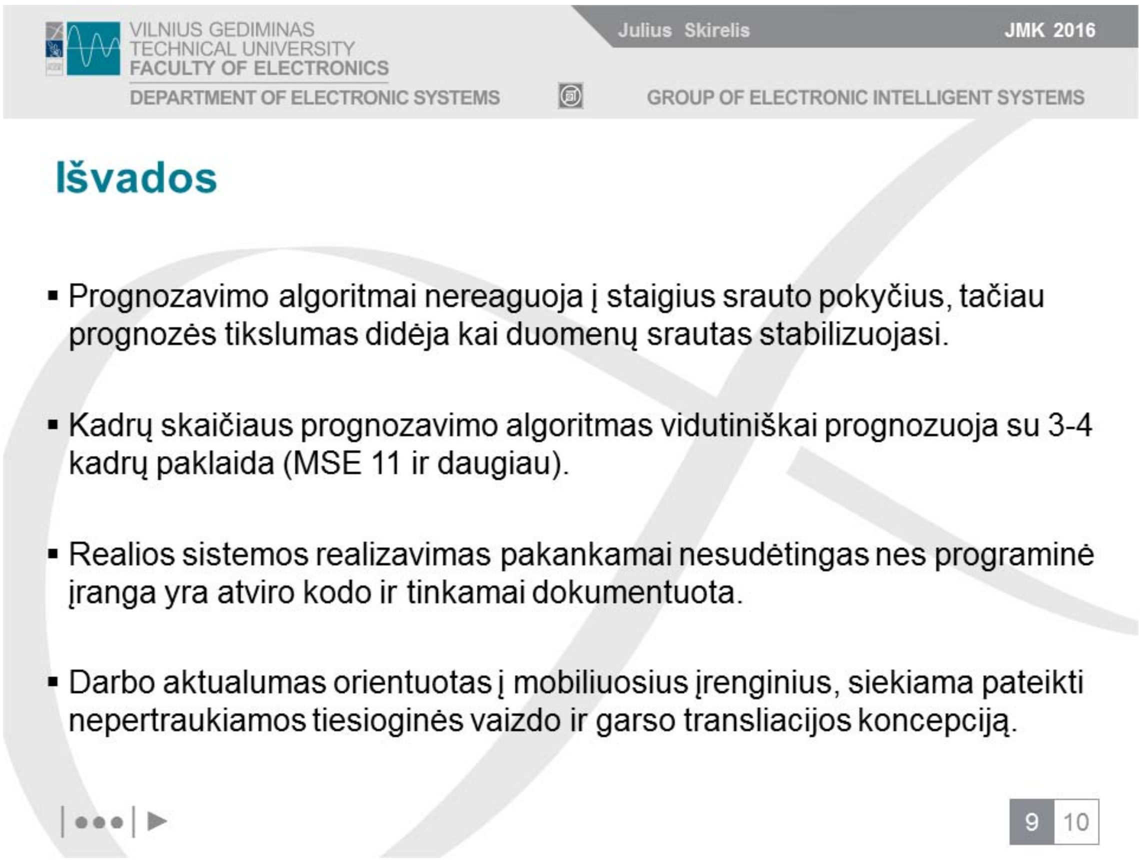
Start Server: http://192.168.1.129:8888

Your name: myclient Connect Disconnect

Clear log

|●●●| ► Slide Title

8 10



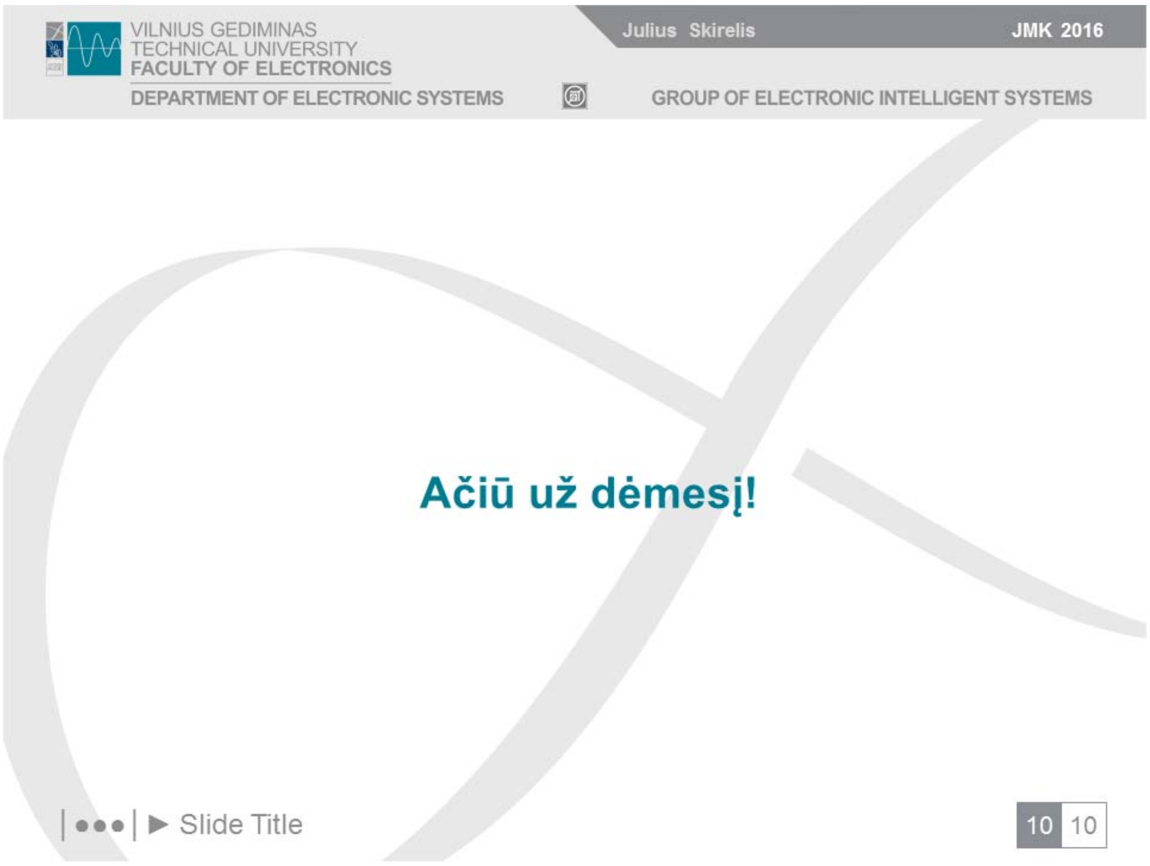
VILNIUS GEDIMINAS TECHNICAL UNIVERSITY FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Julius Skirelis JMK 2016
GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Išvados

- Prognozavimo algoritmai nereaguoja į staigius srauto pokyčius, tačiau prognozės tikslumas didėja kai duomenų srautas stabilizuojasi.
- Kadrų skaičiaus prognozavimo algoritmas vidutiniškai prognozuoja su 3-4 kadrų paklaida (MSE 11 ir daugiau).
- Realios sistemos realizavimas pakankamai nesudėtingas nes programinė įranga yra atviro kodo ir tinkamai dokumentuota.
- Darbo aktualumas orientuotas į mobiliuosius įrenginius, siekiama pateikti nepertraukiamos tiesioginės vaizdo ir garso transliacijos koncepciją.

9 10



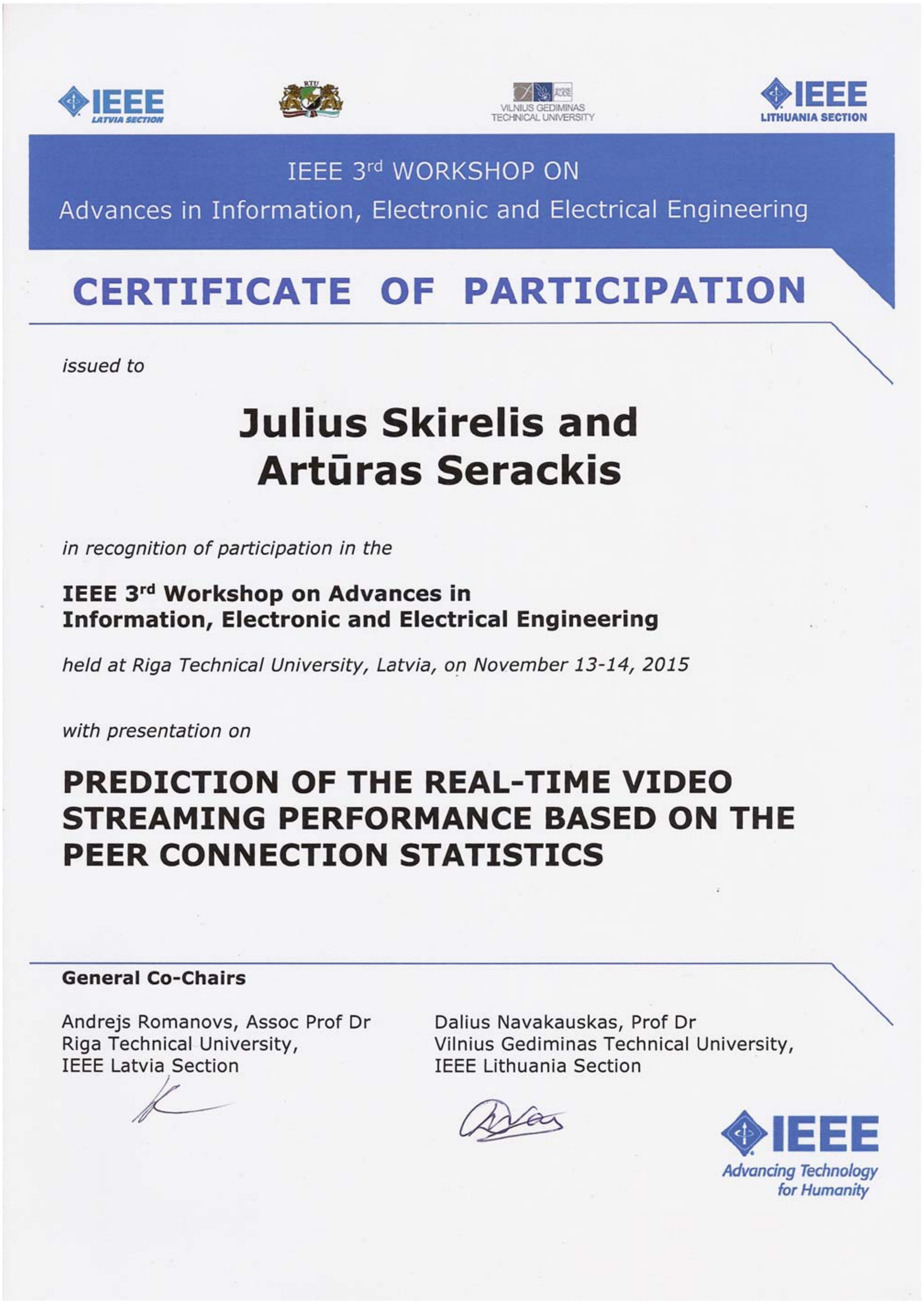
VILNIUS GEDIMINAS TECHNICAL UNIVERSITY FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Julius Skirelis JMK 2016
GROUP OF ELECTRONIC INTELLIGENT SYSTEMS

Ačiū už dėmesį!

▶ Slide Title

10 10



IEEE LATVIA SECTION

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

IEEE LITHUANIA SECTION

IEEE 3rd WORKSHOP ON
Advances in Information, Electronic and Electrical Engineering

CERTIFICATE OF PARTICIPATION

issued to

**Julius Skirelis and
Artūras Serackis**

in recognition of participation in the

**IEEE 3rd Workshop on Advances in
Information, Electronic and Electrical Engineering**

held at Riga Technical University, Latvia, on November 13-14, 2015

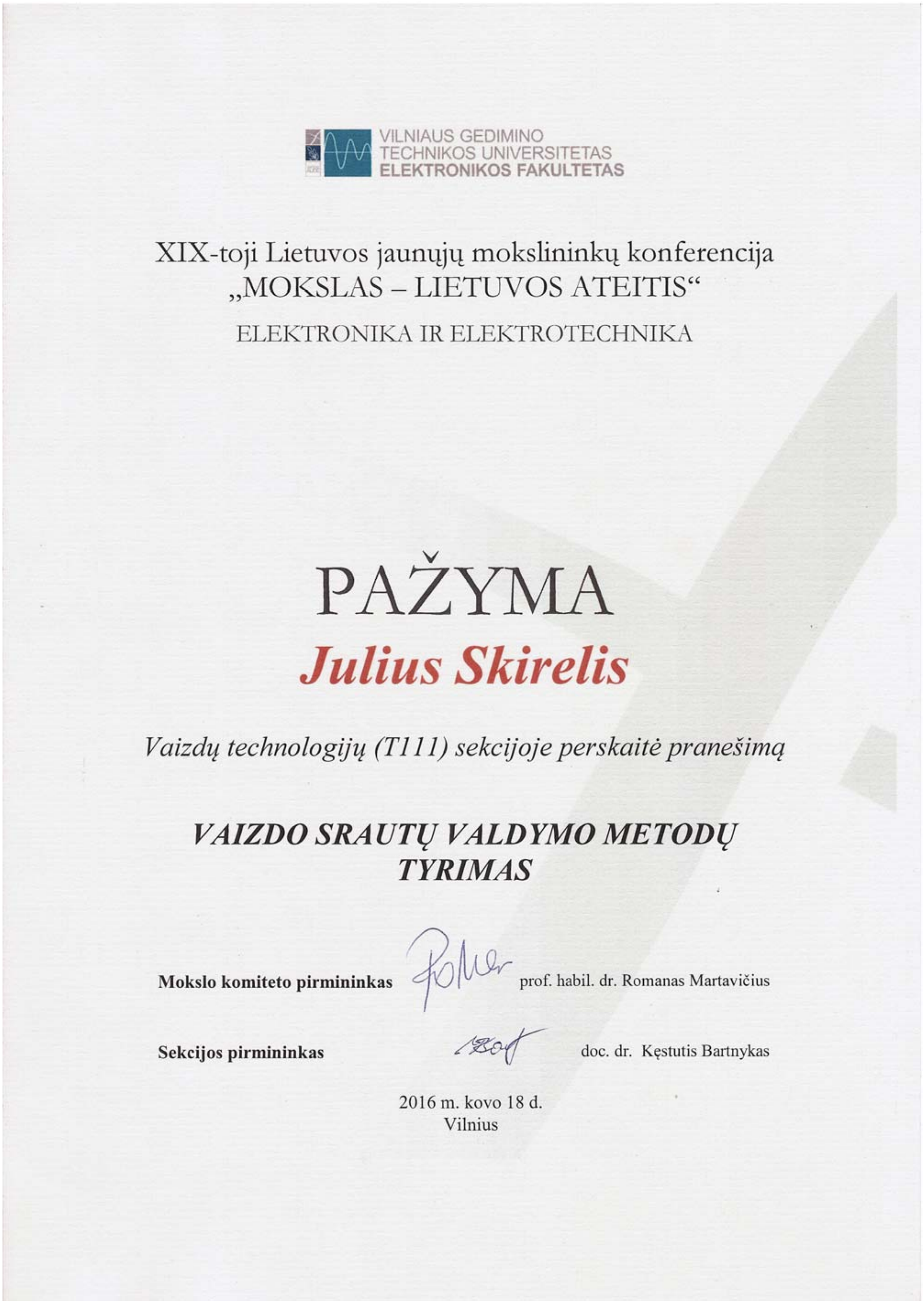
with presentation on

PREDICTION OF THE REAL-TIME VIDEO STREAMING PERFORMANCE BASED ON THE PEER CONNECTION STATISTICS

General Co-Chairs

Andrejs Romanovs, Assoc Prof Dr Riga Technical University, IEEE Latvia Section	Dalius Navakauskas, Prof Dr Vilnius Gediminas Technical University, IEEE Lithuania Section
[illegible]	[illegible]

IEEE Advancing Technology for Humanity



VILNIAUS GEDIMINO
TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS

XIX-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS“
ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA

Julius Skirelis

Vaizdų technologijų (T111) sekcijoje perskaitė pranešimą

VAIZDO SRAUTŲ VALDYMO METODŲ TYRIMAS

Mokslo komiteto pirmininkas [illegible] prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas [illegible] doc. dr. Kęstutis Bartnykas

2016 m. kovo 18 d.
Vilnius