International Journal of Technology (2018) 4: 852-861
ISSN 2086-9614 © IJTech 2018

NEW *ns*-3-BASED EMULATION PLATFORM FOR PERFORMANCE EVALUATION OF TCP-BASED SPEECH RECOGNITION

Kalamullah Ramli [1*], Asril Jarin[2]

[1]*Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Kampus UI Depok, Depok 16424, Indonesia*
[2]*Center of Information and Communication Technology, Agency for the Assessment and Application of Technology (BPPT), Puspiptek Serpong, Tangerang 15314, Indonesia*

(Received: April 2017 / Revised: May 2017 / Accepted: May 2018)

ABSTRACT

Internet-based speech recognition applications prefer using TCP to ensure reliable speech data delivery. TCP-based speech recognition can be designed to push recognition updates on the fly, without waiting for all speech data to fully arrive at the server. We propose an *ns*-3-based emulation platform to evaluate the performance of TCP-based speech recognition. The server and the client are connected to the simulated network using a tap bridge. The real-time performance of full-duplex speech recognition is measured based on data size, loss rate, and propagation delay. For all our data samples, the application exhibits good performance when the propagation delay is 120 seconds and the loss rate is less than 0.3%, as well as when the propagation delay is 50 seconds and the loss rate is less than 0.5%.

Keywords: Emulation platform; ns-3; Simulated network; TCP-based speech recognition; WebSocket

1. INTRODUCTION

In addition to image recognition systems such as face recognition (Shehzad et al., 2014) and character recognition (Abdalkafor, 2017), speech recognition (SR) is an artificial intelligence technology that many research works take the attention today. A speech recognition application recognizes and translates speech into the text with the help of a set of models derived from a "training" or "learning" process of a large training data consisting of utterances that have been aligned with their text. Popular models of speech recognition are Hidden Markov Model, Neural Network, and Deep Neural Network (DNN).

The performance of speech recognition systems in decoding speech data sent over IP networks or the Internet generally degrades due to packet loss and delay (Sciver et al., 2002). Packet loss, in turn, creates errors in recognition processes (Carmona et al., 2010).

TCP is a reliable transport protocol capable of sending data without loss; however, it potentially introduces an unpredictable end-to-end delay. The delay caused by the TCP mechanism may reduce user satisfaction with the speech recognition application while on the other hand, while on the other hand, the performance of this application depends on the availability of data. The faster the data arrive, the faster the recognition results can be provided to the user.

*Corresponding author's email: k.ramli@eng.ui.ac.id, Tel. +62-21-7270078, Fax. +62-21-7270077
Permalink/DOI: https://doi.org/10.14716/ijtech.v9i4.80

Many speech recognition applications wait until all data arrive before starting the recognition process (Peinado & Segura, 2006). However, users' quality of experience (QoE) for online applications demands minimal waiting time. Our previous work investigated acceptable delay in TCP-based speech recognition, especially those using a speech segmenter on the client side (Jarin et al., 2017). For 2500 Indonesian speech sentences, the acceptable sentence delay was found to be 8 seconds when the loss-rate was smaller than 0.05, and the average round-trip time was found to be 100 ms. When the round-trip time was doubled, the acceptable delay was also doubled.

Since the rise of web-based applications recently supported by asynchronous communication technologies, such as *polling*, *long-polling*, and *streaming* (Lubbers & Greco, 2010), TCP-based speech recognition can deliver a real-time service to users by pushing temporary results or updates to the client while the recognition process continues to run on the server. *Polling* is a technique in which the browser periodically sends HTTP requests to the server to synchronize data from the server immediately. The timeliness of polling techniques depends on the synchronization rate between HTTP requests and the availability of data on the server. *Long polling* is a types of polling technique that emulates a server push mechanism. Data are pushed by the server to the client using normal polling, but the server does not need to respond immediately when there are no updates waiting. The server then leaves the request open and awaits the next updates. *Streaming* is a technique in which the browser sends a complete request to the server and the server continuously sends the updates to the client for either an undefined time or a given period of time. The server leaves the connection open to deliver subsequent messages (Lubbers & Greco, 2010).

One form of TCP-based real-time speech recognition application was developed by Alumäe (2014). This application uses the WebSocket protocol, which allows two-way communication between the client and the server (Fette & Melnikov, 2011). Using WebSocket, the client sends its requests and speech data to the server in any container and using any encoding supported by the GStreamer framework (https://gstreamer.freedesktop.org). The server then sends back to the client the intermediate recognition results in the form of JavaScript Object Notation (JSON) code. The server employs the Kaldi online decoder to segment incoming speech into speech sentences in real time. For each sentence, the decoder produces progressive hypotheses and then one final hypothesis (Povey et al., 2011).

The development of a TCP-based speech recognition system requires a test to evaluate performance. Studies with limited funds that are not part of a global research network often conduct evaluations using network simulators. Network simulators help to minimize costs; however, the accuracy of their results is limited (Alvarez et al., 2010).

We propose an *ns*-3-based emulation platform to evaluate the performance of TCP-based speech recognition. The platform is designed to run on a single machine and consists of three main parts: a speech recognition server running processes in a local host, a client running an Internet browser on a virtual machine, and an *ns*-3-based simulated network running another process on the local host. Both the server and the client are connected by a tap-bridge to the simulated network. The client sends speech data over an emulated Internet (i.e. a dumbbell topology in *ns*-3), and the server sends back to the client the intermediary results of the speech recognition as server updates. We validate the platform using a full duplex speech recognition application developed by Alumäe (2014). We also identify the working region of the application's real-time service.

Based on our literature review, our platform differs from recent platforms using *ns*-3-based simulated networks. Whereas recent platforms, such as those proposed by Fouda et al. (2014)

and Molloy et al. (2014), have focused on testing and emulating long-term evolution (LTE) technology, our platform evaluates TCP-based speech recognition.

The working region of the real-time application is identified using the parameters of speech length, loss rate, and propagation delay. Good performance is achieved when the real-time factor (RTF) is smaller than one, as specified by Platek (2014).

The remainder of this paper is organized as follows. Section 2 presents an overview of how to make *ns*-3 interact with the real world and the tools used to read the measurement results. Section 3 describes the *ns*-3-based emulation platform and utilizes it in an experiment to evaluate the real-time services of TCP-based speech recognition over *ns*-3. In Section 4, the results of the experiment are discussed, and conclusions of the research are presented. Finally, Section 5 concludes the paper.

2. METHODS

2.1. TCP-based Speech Recognition

Implementations of speech recognition systems on the Internet or public networks experience degradation due to packet loss and data delay. Applications generally prefer to delay until all speech data are available. Such data availability supports improved speech recognition accuracy. The TCP protocol guarantees the delivery of all data from the client to the server with certain delays. Today, many TCP-based applications anticipate this delay by employing asynchronous communication techniques, such as polling, long polling, server-sent events (SSE), and WebSocket (Pimentel & Nickerson, 2012; Grigorik, 2013). In our other work, a framework using the HTTP/2 protocol plus Server Sent Event (SSE) provides a better solution to the timeliness of TCP-based speech recognition application. HTTP/2 is an application protocol published by IETF in May 2015 and aims to reduce latency at the application level (Belshe et al., 2015).

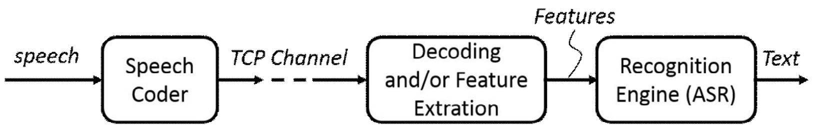


Figure 1 TCP-based speech recognition scheme

As illustrated in Figure 1, TCP-based speech recognition systems send speech data over a TCP channel. The speech data are compressed and can be sent by a client directly from the speaker or from files in certain formats (e.g. WAV, MP3, raw, etc.).

2.2. Developing Real Applications over *ns*-3

Experimentation over a conditioned Internet environment is necessary to measure the latency and the real-time factor (RTF) of TCP-based speech recognition services. In real life, such experiments are very difficult and expensive to realize outside of global research networks, such as PlanetLab (http://www.planetlab.org).

To run the simulation, *ns*-3 abstracts the nodes containing applications, stacks and network devices, and channels. Every application contributes to generating traffic, and the network devices move the traffic through the channels (https://www.nsnam.org). The physicality of the network and the nodes is classified as either a real or a simulated network. Based on the work of Dowell (2010), a configuration with tap devices is used to make real nodes interact with *ns*-3.

Simulations use two types of virtualization platforms: full virtualization and paravirtualization. In full virtualization, a virtual machine environment is created to completely simulate the underlying hardware and operating system (OS) environment. Using this platform, one can make a single system appear to be multiple systems of the same or different types. Popular examples of full virtualization systems include VMWare (https://www.vmware.com) and VirtualBox (https://www.virtualbox.org). Full virtualization environments that simulate all aspects of the system can be heavy, and individual virtual machines must be isolated from one another. In paravirtualization, by contrast, a virtual machine environment is created, but the underlying machine is not completely simulated.

In our proposed platform (Figure 3), we create a full virtualization environment for the client, while the server and the simulated *ns*-3 network run on the Linux host. To allow *ns*-3 to work with the virtualization system, the platform uses a tap device mechanism. The tap device is a virtual network interface (i.e. a software) created using *Tunctl Linux Package* (Zimmerman, 2013). It simulates a link layer device and works with layer-2 packets like Ethernet frames. In *ns*-3, a net device called a 'tap bridge' is installed on a ghost node to connect the tap device with a simulated network. The tap bridge acts like an extension of a Linux bridge (`brctl`) that connects *ns*-3 to a tap device created by the Linux host. Overall, *ns*-3 provides three configuration modes for implementing tap bridges to connect to tap devices: *ConfigureLocal*, *UseLocal*, and *UseBridge* (NS-3, 2011).

2.3. Utilizing Platform for Performance Evaluation

To validate and use the platform to evaluate performance, we design the experiment steps as follows:

(1) Configure the tap devices for the client and server nodes.
- Create the tap devices for the client-side interfaces (`tap0` and `tap2`) and the server-side interface (`tap1)` on the Linux terminal using the `tunctl` command.
- Create the Linux-bridge for bridging the tap devices (`tap0` and `tap2`) on the client side using the `brctl` command.
- Configure and enable the Linux-bridge and tap devices on the client side (`tap0` and `tap2`) using IP address `0.0.0.0`.
- Configure the tap device for the server-side interface using IP address `10.2.1.250`. This IP address must be registered as a server address in the simulated network.

(2) Run the server for the full duplex application (master server and worker) on the local host (Linux).

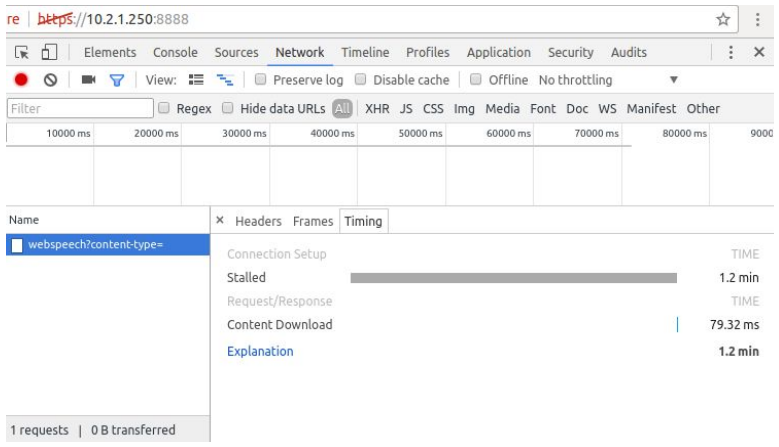
(3) Run the virtual machine for executing the application client (e.g. Internet browser). The tap device, which is connected to the virtual machine, is previously set for `tap0`. The virtual interface of the client is configured with the IP address `10.1.1.254`. This IP address must be registered as a client address in the simulated network.

(4) Run the simulated *ns*-3 network on the local host (Linux).

Each of the above running processes runs on a separate terminal.

2.4. Presenting the Measurement Results

Several tools can be used to capture the experiment results and to obtain targeted measurements. Using *Wireshark*, various aspects of packet traffic (e.g. protocol information, packet delay, loss rate, etc.) can be inspected and analyzed. Other useful tools are available from Google Chrome (i.e. `chrome://net-internal)` and other developer tool sources. Figure 2 shows a timing measurement of WebSocket communication captured using Google Chrome's developer tools.


re | ~~https~~://10.2.1.250:8888
Elements Console Sources Network Timeline Profiles Application Security Audits
View:
Preserve log
Disable cache
Offline
No throttling
Filter
Regex
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
10000 ms 20000 ms 30000 ms 40000 ms 50000 ms 60000 ms 70000 ms 80000 ms 9000
Name
Headers Frames Timing
1 requests | 0 B transferred

webspeech?content-type=

Connection Setup	TIME
Stalled	1.2 min
Request/Response	TIME
Content Download	79.32 ms
Explanation	1.2 min

Figure 2 Timing measurement of full-duplex service using Google Chrome developer tools

3. RESULTS

This work yields two results. One is an ns-3-based emulation platform for evaluating the performance of TCP-based speech recognition. The other is the experimental results of the evaluation of a TCP-based full-duplex recognition application using this platform.

3.1. ns-3-based Emulation Platform

By employing tap devices in coordination with *ns*-3 tap bridges, our emulation platform evaluates the real-timeness of applications running over a simulated *ns*-3 network. The tap bridge works to integrate "real" internet hosts into *ns*-3 simulations. As shown in Figure 3, the platform is composed of three main parts: a real node for the speech recognition client, a real node for the speech recognition server, and a simulated *ns*-3 network.

3.1.1. Speech recognition client (Virtual Host)

The speech recognition client, which is executed in an Internet browser, runs on a Virtual Linux Host. The Virtual Linux Host is a Linux guest system virtualized by a virtual machine, such as Virtual Box or VMWare. The client is connected to the simulated *ns*-3 network using the *UseBridge* configuration, in which an *ns*-3 tap bridge logically extends the Linux Bridge (`brctl`) into *ns*-3. The Linux Bridge connects two tap devices (`tap0` and `tap2`) to support communication among the virtual machine (Linux guest system), the Linux host, and the simulated *ns*-3 network. Tap device `tap0` makes the client in the virtual machine appear as a node, which is connected by the Linux Bridge into the simulated *ns*-3 network. Meanwhile, tap device `tap2` makes the simulated *ns*-3 network appear in the Linux host. Each tap device is provided and configured by the Linux host and connected to an *ns*-3 net device via a tap bridge.

3.1.2. Speech recognition server (Local Host)

The speech recognition server is the real node that provides real-time speech recognition services. The server runs on a local host (Linux) and serves the client request over the *ns*-3 channel. The server is connected to the simulated *ns*-3 network using the *UseLocal* configuration. The *ns*-3 tap bridge uses tap device `tap1`, which is previously created and configured on the local host on which the server running.

3.1.3. Simulated network (Local Host)

The simulated network is designed using a *dumbbell* topology because this is the topology typically used to analyze TCP-based services. The dumbbell network represents the Internet in the *ns*-3 channel. The topology can be created via multiple TCP and HTTP sources connected to a bottleneck router on the left side (r_0), with corresponding sinks connected to another bottleneck router on the right side (r_1). Each HTTP source can be set to have multiple connections, and their traffic can be generated on demand.

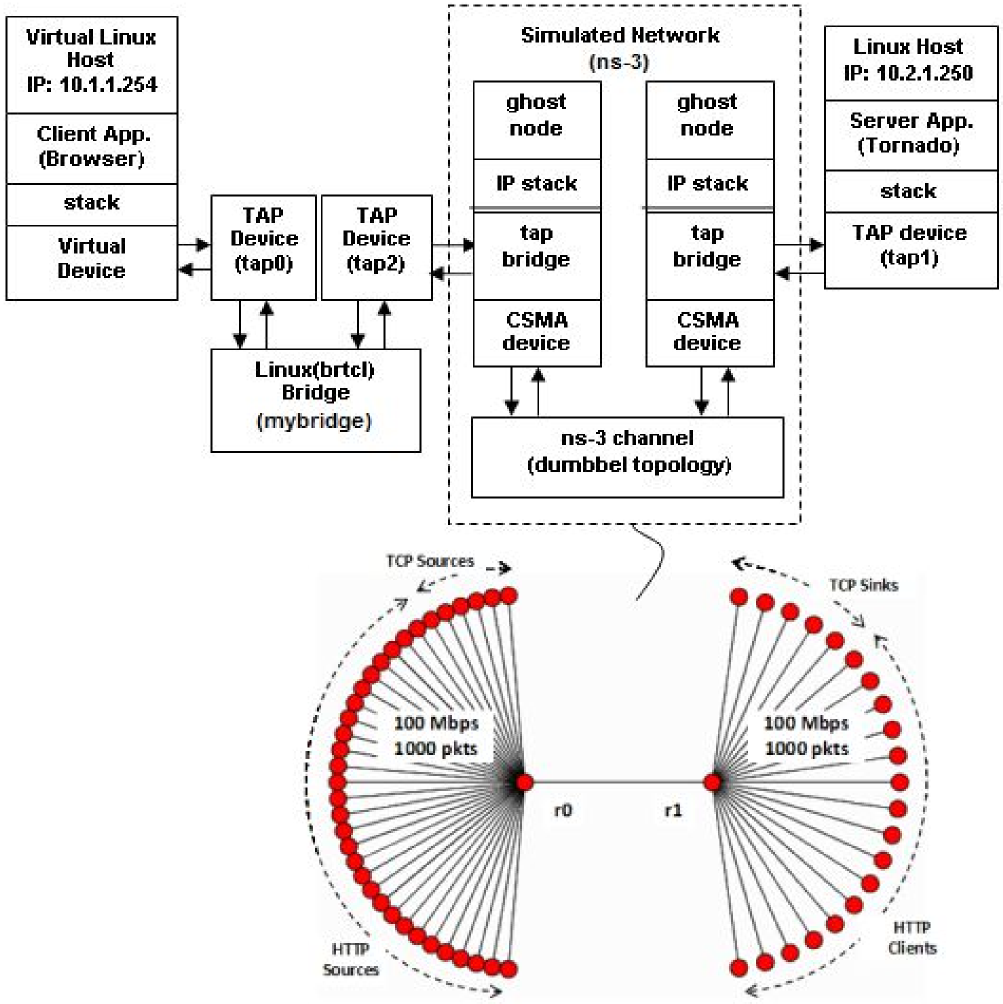


Figure 3 Emulation platform for TCP-based speech recognition over *ns*-3

As shown in Figure 3, an *ns*-3 ghost node represents the connected real node. In this platform, the ns-3 CSMA devices are interfaces implemented to interact with the real nodes of the speech recognition application using the tap and tap-bridge configuration. On the client side, they are configured in *UseBridge* mode, and on the server side, they are configured in *UseLocal* mode.

3.2. Experiment

To evaluate the performance of TCP-based speech recognition, an experiment was conducted following the steps described in section 2.3. Alumae's (2014) full duplex speech recognition application was used. The latency and RTF of the application were measured based on certain settings and samples of speech data. The RTF measures the real-timeness of speech recognition based on the ratio of decoding time length to speech length. An application has real-time service when the RTF value is smaller than one (Platek, 2014).

Furthermore, we used the dumbbell setting (Jarin et al., 2017), in which multiple TCP and HTTP sources were connected to *r0*, and their corresponding sinks were connected to router *r1*. Each HTTP source contained 16 connections. One of the TCP sources became a ghost node of the speech recognition client, and one of the TCP sinks became a ghost node of the speech recognition server. Traffic from other sources, both HTTP and TCP, was generated using empirical data provided by *ns*-3. The bandwidth and queue length of the link from a source/sink to its corresponding router were 100 Mbps and 1000 packets, respectively. The propagation delay of the link from a source/sink to its corresponding router was uniformly 20 ms. The round-trip propagation delay was equal to twice the propagation delay of the links between the source and the sink.

We also determined four parameter settings for *ns*-3 simulations, as listed in Table 1.

Table 1 Four parameter settings of the simulated *ns*-3 network

Setting	# of Sources		Bottleneck Link			Parameters of Application		
	TCP	HTTP	Prop. Delay (ms)	Bandwidth (Mbps)	Buffer (pkts)	Round trip Prop. Delay (ms)	Loss Rate	Average RTT (ms)
1	9	40	40	3.7	50	120	0.4% - 0.5%	95
2	5	30	40	3.7	50	120	0.2% - 0.3%	87
3	9	40	5	5	100	50	0.4% - 0.5%	95
4	5	30	5	5	100	50	0.2% - 0.3%	87

The full-duplex application was modified in such a way that it could be accessed using an Internet browser (Alumäe, 2014). To measure latency and RTF, we used ten different speech data samples, as listed in Table 2. The data lengths were not linear; thus, they did not ensure the linearity of experiment results.

Table 2 Ten speech data samples used for the experiment

No.	Speech Data	Size (MB)	Length (sec)
1	Track05.wav	11.6	66.05
2	Track03.wav	11.8	67.03
3	Track21.wav	13.1	74.15
4	Track11.wav	13.5	76.31
5	Track12.wav	13.9	78.83
6	Track06.wav	14.6	82.67
7	Track02.wav	15.1	85.61
8	Track01.wav	15.9	90.43
9	Track04.wav	16.2	91.71
10	Track24.wav	19.2	109.01

The results were then used to compare the application latencies and RTFs of the ten speech data within four experiment settings, as shown in Figures 4 and 5.

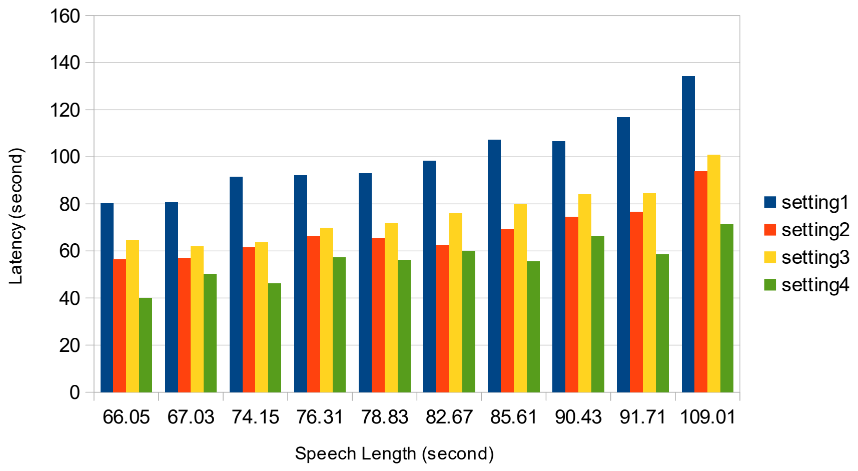


Figure 4 Latency comparison of each experiment setting

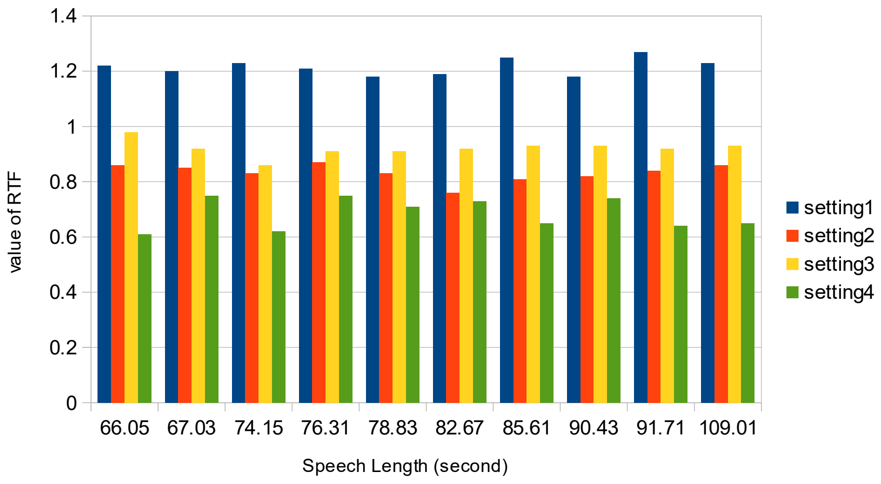


Figure 5 RTF comparison of each experiment setting

4. DISCUSSION

The main parameters of evaluation in this experiment were application latency and RTF, as these are the determinants of TCP's success in providing speech recognition applications that ensure user satisfaction.

Given the assumption that the computing power of each experiment is constant, the measurement results are diverse due to variations in speech length (or data size), loss rates, and propagation delays. Settings 1 and 3 produce roughly the same loss rate, but have different propagation delays. The propagation delay in setting 1 is higher than that in setting 3. Conversely, settings 2 and 4 have smaller loss rates than settings 1 and 3. Overall, these results suggest the following conclusions:

- Application latency is proportional to the increase in speech length, loss rate, and propagation delay.
- Application latency is mostly influenced by propagation delay. Based on the results shown in Figure 5, setting 1 yields an RTF with a value greater than one.
- Latencies caused by loss rate and propagation delay are more significant than latencies caused by speech decoding on the server.

Figures 4 and 5 show the latency and RTF of each experiment setting with different round-trip delays and loss rates. We found that the full-duplex application exhibits good performance for all speech data when the round-trip propagation delay is 120 seconds and the loss rate is smaller than 0.3% (settings 2 and 4) or when the round-trip propagation delay is 50 seconds and the loss-rate is smaller than 0.5% (settings 1 and 3).

5. CONCLUSION

This paper proposes an *ns*-3-based emulation platform to evaluate the real-timeness service of TCP-based speech recognition. The platform consists of three main parts: a client, a server, and a simulated *ns*-3 network. All parts run on a single machine. The real application of speech recognition is connected using tap devices and an *ns*-3 tap bridge configuration.

We performed an experiment to evaluate the full-duplex speech recognition and identified the working region in which the application may achieve real-timeness. Specifically, the application exhibits good performance when the propagation delay is 120 seconds and the loss rate is less than 0.3% and when the propagation delay is 50 seconds and the loss rate is less than 0.5%.

6. ACKNOWLEDGEMENT

This article's publication is supported by the United States Agency for International Development (USAID) through the Sustainable Higher Education Research Alliance (SHERA) Program for Universitas Indonesia's Scientific Modeling, Application, Research and Training for City-centered Innovation and Technology (SMART CITY) Project, Grant #AID-497-A-1600004, Sub-grant #IIE-00000078-UI-1.

7. REFERENCES

Abdalkafor, A. S., 2017. Designing Offline Arabic Handwritten Isolated Character Recognition System using Artificial Neural Network Approach. *International Journal of Technology*, Volume 8(3), pp. 528-538

Alvarez, A., Orea, R., Cabrero, S., Paneda, X.G., Garcia, R., Melendi, D., 2010. Limitations of Network Emulation with Single-machine and Distributed ns-3. *In:* Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, pp. 1–9

Alumäe, T., 2014. Full-duplex Speech-to-text System for Estonian. *In:* International Conference Human Language Technologies—the Baltic Perspective, Kaunas, Lithuania, pp. 3–10

Belshe, M., Peon, R., Thomson, M., 2015. RFC 7540. *Hypertext Transfer Protocol Version 2 (HTTP/2)*

Carmona, J.L., Peinado, A.M., P'erez-C'ordoba, J.L., 2010. Coded-speech Recognition over IP Networks. *In:* Proceedings of FALA 2010-VI Jornadas en Tecnologia del Habla and II Iberian SL tech Workshop, pp. 287–290

Dowell, C., 2010. HOWTO Make ns-3 Interact with the Real World. Available online at https://www.nsnam.org/wiki/HOWTO_make_ns-3_interact_with_the_real_world, Accessed on 4 April, 2017

Fette, I., Melnikov, A., 2011. *The WebSocket Protocol*. RFC 6455

Fouda, A., Ragab, A.N., Esswie, A., Marzban, M., Naser, A., Rehan, M., Ibrahim, A.S., 2014. Real Time Video Streaming Over ns-3-based Emulated LTE Networks. *International Journal of Electronics Communication and Computer Technology*, Volume 4(3), pp. 659–663

Grigorik, I., 2013. *High Performance Browser Networking: What Every Web Developer Should Know About Networking and Web Performance*. O'Reilly Media Inc., Sebastopol, Canada

GStreamer, 2017. GStreamer Open Source Multimedia Framework. Available online at https://gstreamer.freedesktop.org, Accessed on 4 April, 2017

Jarin, A., Fahmi, H., Suryadi, S., Ramli, K., 2017. Development of Modified Analytical Model for Investigating Acceptable Delay of TCP-based Speech Recognition. *Advanced Science Letters*, Volume 23(4), pp. 3654–3659

Lubbers, P., Greco, F., 2010. HTML5 WebSocket: A Quantum Leap in Scalability for the Web. Available online at https://www.websocket.org/quantum.html, Accessed on 4 April, 2017

Molloy, T., Zhenhui, Y., Muntean, G.M., 2014. Real Time Emulation of an LTE Network Using ns-3. *In:* Proceedings of the 25th IET Irish Signals and Systems Conference 2014 and China-Ireland International Conference 2014, pp. 251–257

NS3-Consortium, 2011. Tap Bridge Model [Devices]. Available online at https://www.nsnam.org/docs/release/3.9/doxygen/group__tap_bridge_model.html, Accessed on 4 April, 2017

NS3-Consortium, 2015. NS-3 Network Simulator. Available online at https://www.nsnam.org, Accessed on 4 April, 2017

Peinado, A., Segura, J.C., 2006. *Speech Recognition over Digital Channels: Robustness and Standards*. John Wiley & Sons Ltd., Chichester, England

Pimentel, V., Nickerson, B.G., 2012. Communicating and Displaying Real-time Data with WebSocket. *IEEE Internet Computing*, Volume 16, pp. 45–53

Platek, O., 2014. Automatic Speech Recognition using KALDI. *Master's Thesis*, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., 2011. *The Kaldi Speech Recognition Toolkit*. Available online at http://kaldi.sourceforge.net, Accessed on 4 April, 2017

Sciver, J.V., Ma, J.Z., Vanpoucke, F., Hamme, H.V., 2002. Investigation of Speech Recognition over IP Channels. *In:* Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Orlando, Florida, USA

Shehzad, M. I., Awais, M., Amin, M., Shah, Y. A., 2014. Face Recognition using Average Half Face Template. *International Journal of Technology*, Volume 5(2), pp. 159-168

VMware, 2017. VMware Virtualization for Desktop, Server, Application, Public and Hybrid Cloud. Available online at https://www.vmware.com, Accessed on 4 April, 2017

VirtualBox, 2017. Oracle VM VirtualBox. Available online at https://www.virtualbox.org, Accessed on 4 April, 2017

Zimmerman, M., 2013. Linux Manual Page. Available online at https://linux.die.net/man/8/tunctl, Accessed on 4 April, 2017