TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)
31 Oct - 3 Nov 2023. Chiang Mai, Thailand

Offline Collaboration Tool utilizing WebRTC in Ad Hoc Peer-to-Peer Networks

Kiefer Micco J. Victoriano, Juan Carlo M. Santos, Tingcap B. Mortaba II, Md. Jassim Caballes, and Jaybie A. de Guzman
Smart Systems Laboratory, Electrical and Electronics Engineering Institute, University of the Philippines

Abstract—**Advancement in technology introduced us to various platforms that changed our way of communicating and collaborating. With the sudden demand for technological dependence, the COVID-19 situation exposed the limited internet infrastructure and costly internet service of countries like the Philippines. In this study, the researchers explored the option of real-time offline collaboration in an ad hoc network as a solution to unimpeded collaboration in low bandwidth and poor connectivity environments. The researchers developed on top of CONCLAVE, an open-source collaborative text editor that utilizes WebRTC. Offline functionality was enabled in this collaborative platform along with the implementation of offline peer discovery using Websockets and a peer handling technique inspired by Scalable Content Addressable Networks (CAN). These modifications were evaluated on an ad hoc network using different metrics such as packet delay, jitter, data rate, and peer disconnection resolution time. Results of the testing was favorable to the modified version of CONCLAVE in an ad hoc network. With these results, the project has successfully enabled offline use and continued operation in the event of a disconnected peer, and has successfully implemented a peer handling technique that has improved the P2P operation of the original application.**

I. INTRODUCTION

Technology has changed the way we communicate and interact with each other: enabling us to connect with anybody around the world. Specifically, numerous organizations have transition towards a fully digital and online set-up. And with the emergence of the COVID-19 global pandemic in the past years, the reliance on digitalization has become more salient. It signaled the increased popularity of different collaboration tools, especially now in the pandemic era where we were and still are forced to socially distance ourselves from one another. This sudden demand for technological dependence has exposed the limited internet infrastructure and costly internet service of countries like the Philippines.

Thus, in this work, we enabled *offline functionality* of CONCLAVE, a collaborative web application built utilizing WebRTC [1], and improved its overall performance by modifying its existing peer handling. Our specific contributions are as follows:

1) Enabled offline mode to Conclave's real-time collaboration;
2) Demonstrated peer-to-peer handling characterized by a balanced topology that can scale efficiently;
3) Allowed for continuous operation in the event of a crashed/disconnected host; and
4) Evaluate the P2P implementation based on the following performance metrics: a) *delay*, b) *data rate*, c) *jitter*, and d) *peer disconnection resolution time*.

II. RELATED WORK

A. Collaboration Tools

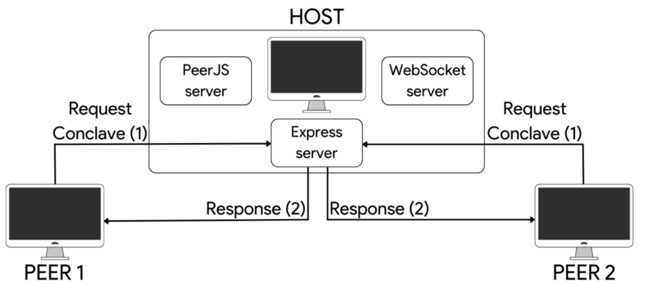
Collaborative tools have gained popularity especially during the COVID-19 pandemic [2]. Google Docs and MS Office Online offer a platform for real-time document editing for multiple authors. These, however, are limited by the availability of an internet connection. [3]. Wooki enables P2P collaborative writing. However, it is also reliant on a constant internet connection to function [4]. Collabio is a collaboration tool that has offline functionality, making use of an ad hoc P2P network. Accessibility is a problem for this application since it is only available in Apple operating systems, requires a paid subscription, and is not open-source [5].

B. Peer-to-Peer Overlay Network

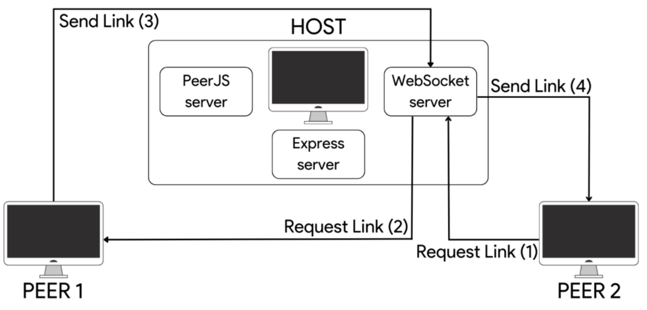
CONCLAVE operates in a decentralized manner where each peer maintains an up-to-date local copy of a document. And with every change, each peer is responsible for relaying these changes to all other peers in order to maintain consistency between the local copies. Topology and peer management of the network threaten the reliability and quality of service that CONCLAVE can deliver [6]. In the subsequent sections, various related work on the approaches to enabling P2P connectivity is discussed.

1) Peer-to-Peer Architectures: P2P overlay networks, which can be classified as either structured or unstructured P2P networks, are decentralized networks of peers where each serve as both a router and a host device. Unstructured networks lack a defined process for building and maintaining the network, which makes fault tolerance uncertain. Structured networks, however, have specialized nodes that make queries more efficient. *Dynamic Hash Tables* (DHTs) offer a structured mapping of peer IDs to values and store location data of peers in the network for routing and address purposes which addresses specific challenges in P2P networks [7]. Example topologies of various structured P2P networks include Chord, Kademlia, Tapestry, Pastry, Content Addressable Networks (or CAN), and Viceroy. Out of all these approaches, CAN is the simplest to implement.

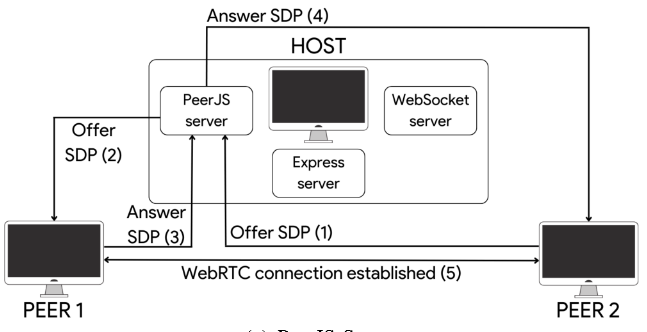
2) Mobile Ad Hoc Networks: Mobile Ad Hoc Networks (MANETs) are wireless networks that do not rely on any infrastructure, and instead, each device in the network acts as both a router and an end device [8]. Some of the commonly used routing protocols in MANETs are DSDV, CGSR, and WRP [9]. MANETs are wireless, infrastructure-free, and dynamic networks that offers resistance to abrupt changes in the network's topology. Techniques and topologies found in



(a) Express Server



(b) WebSocket Server



(c) PeerJS Server

Fig. 1. Services enabling offline collaboration

MANETs can be considered for improving the peer-to-peer topology management of CONCLAVE. While CONCLAVE's peer-to-peer topology is a logical overlay network on top of an infrastructure-based network, MANETs are physical networks.

III. ENABLING OFFLINE COLLABORATION IN CONCLAVE

CONCLAVE is an open-source web application built mainly on JAVASCRIPT/NODE.JS, and serves as the main foundation for building the offline collaboration tool. It uses WebRTC for real-time communication. With its original implementation, the host-peer acts as a temporary central server until there are five peer connections, at which point it directs new connections to other peers in the network. This achieves the majority of the desired goals except for offline functionality and better peer-to-peer network handling[1]. To handle collaborative editing, CONCLAVE uses *conflict-free replicated data type* (CRDT). CRDT assigns a global unique identifier to every character and utilizes a counter to track the number of operations performed to ensure matching between replicas [10].

The modified application architecture took advantage of two layers, the `connecting` layer and the `application` layer. The `connecting` layer is where all the prerequisite services are established to enable the collaboration. An initial client-server model in the form of the `connecting` layer is necessary as one peer was designated a the host or *bootstrap* peer. Beyond the `connecting` layer, the `application` layer operates fully peer-to-peer. At the `application` layer, all peers in the overlay network bear identical responsibilities and abilities, making the collaboration and administration of the network completely peer-to-peer.

A. Peer Discovery

1) Hosting: As a web application, CONCLAVE relies on a server that hosts the assets necessary to build the application on the client side. For this to work in an offline environment, Express, a NODE.JS framework was used. As shown in Fig. 1.a, an Express server handles HTTP requests and responses. It simplifies the process of handling HTTP requests and responses by providing a high-level API for defining routes, handling requests, and sending responses. Once the server is up, other users can now request the offline CONCLAVE application using the IP address of the host peer and the respective port.

2) Connecting: Originally, CONCLAVE facilitates peer discovery by generating an invite link that a user can send to other users. Sharing the invite link is usually convenient in an online environment where you can easily use applications like Messenger or Discord. In an offline environment, this method is not a practical option anymore.

To address this, as shown in Fig. 1.b, the WebSocket protocol is utilized using NODE.JS. A local WebSocket server is initialized on port 8886 on the host peer. This is instantiated along with the Express server that the offline CONCLAVE runs on. Once a CONCLAVE instance is opened, it will automatically connect to this server. This connecting functionality was inspired from the open-source project, Snapdrop [11].

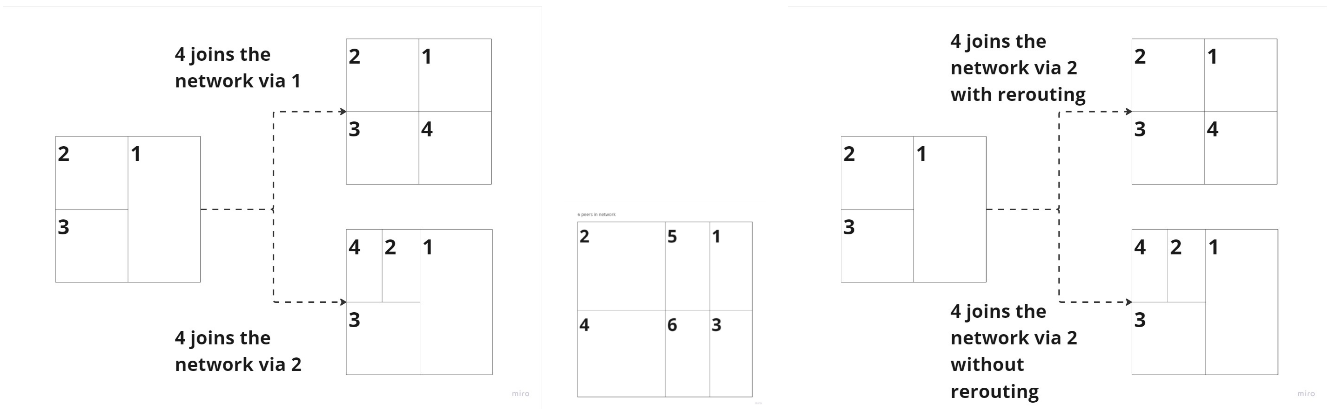
3) Establishing WebRTC Connection: CONCLAVE uses PeerJS, a WebRTC framework. Signaling is a requirement in connecting two WebRTC agents. Signaling is done to exchange session description protocol (SDP) of two different peers, necessary for peers to know each other's location in a network to initiate data exchange.

The original CONCLAVE application uses an online PeerJS server as a signaling server. For CONCLAVE to work in an offline environment, as shown in Fig. 1.c, a local PeerJS server was used instead using the `peer` package, the PeerJS server component offered by NODE.JS. The local PeerJS server was instantiated and binded on the same port as the Express server mentioned in Section III-A1.

B. Peer-to-Peer Handling

The modified P2P handling integrates CAN functionalities [12] [13] into the modified CONCLAVE implementation.

1) Bootstrapping: Bootstrapping is responsible for the initiation of new peers into the CONCLAVE application, and enables the functionalities necessary for connecting them to the application. Bootstrapping here, in particular, refers to bootstrapping into the application layer network which is separate from the bootstrapping of peers in the connecting


4 joins the network via 1
2
1
3
4
2
1
3
4
2
1
3
4 joins the network via 2
2
5
1
4
6
3
4 joins the network via 2 with rerouting
2
1
3
4
2
1
3
4
2
1
3
4 joins the network via 2 without rerouting
miro

(a) Zone Splitting Convention (b) Six Peer Network (c) Four Peer Network With and Without Rerouting

Fig. 2. Zones and rerouting in the Peer Network

layer. It follows that the term 'network' in the succeeding parts refer to the CAN.

Bootstrapping into an existing CAN would require a peer to connect to a specific peer's instance, and effectively its network, done through the utilization of an assigned PeerJS `peerID`. CONCLAVE instances are run on each end device and do not require a central host to operate. Likewise, the network is, necessarily, not reliant on a central peer to facilitate the collaboration.

In the original CONCLAVE application, a sharing link is used to invite peers into a network. This would not be viable for an offline implementation. In the modified version, every instance within the ad hoc network can discover other instances in the network and are given the option to connect to any instance in the network.

2) Zone Splitting: Zone splitting involves the construction of the CAN and allocation of 'zones' to connecting peers. Upon joining a CONCLAVE network, the connecting peer is assigned half of the target peer's original zone. A *zone* is a set of characteristics that describe the area that a peer occupies in a theoretical cartesian space. These characteristics are the region, generation value, neighbors, and etc. These are modified when a zone split occurs. The zone splitting convention is illustrated in Fig. 2.a.

3) Connecting: Connecting is responsible for establishing connections to other peers in the network and reorganizing connections with respect to the network architecture. Each peer has a specific zone within the Cartesian space, and peers with zones that are abut each other are considered neighboring zones. In the network, only neighbor zones will form P2P connections with each other in order to maintain the CAN architecture. For example, in Figure 2.b, peer 5 only has knowledge of peers 1, 2, and 6, and will only form P2P connections with these peers.

4) Rerouting: Rerouting is a process primarily focused on balancing the network by forwarding peers to neighbors with a larger zone to redistribute zone allocation. Since peers can freely connect to any instance, rerouting is implemented to reduce oversaturation of peers in an area of the cartesian space. Essentially, the connection request made by the connecting peer to the target peer is forwarded its neighbor with a larger zone. If the target peer fails to find a larger zone, it accepts the connection request and continues with the connecting process. An example is seen in Fig. 2.c.

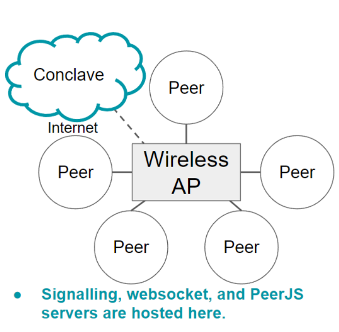
5) Disconnecting: Disconnecting is responsible for seamless and proper zone/resource reallocation in the network when a peer disconnects from the network. CONCLAVE's original implementation has a branching star topology, giving rise to significant issues. Particularly, in these applications, version consistency is important. One of the identified problems with the current P2P handling mechanism is partitioning and network recovery. That is, a disconnecting peer can cause a portion of the network to be partitioned from the rest of the network.

Additionally, without a central list of existing peers in the network, network changes are not streamlined. This is due to each peer having different views of the network, and could only operate within that localized view. To keep track of alive peers in the network, the concept of a *heartbeat* message is used. A heartbeat protocol is a method used to determine the state of machines in a decentralized network [14]. This is done by periodically sending a control message, also known as a heartbeat message, to a peer in the network. The recipient peer then replies to the sender to verify its state and initiates the recovery when a neighbor fails to send a heartbeat message.

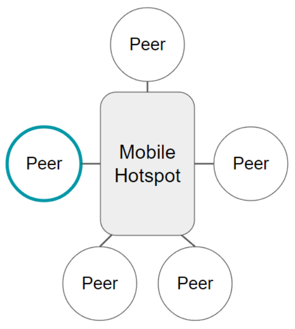
When a peer disconnects, it leaves behind its zone. Without a recovery process in place, this resource is left unusable to any peer in the network. In CAN, the cartesian space becomes fragmented, disrupting the topology of the network rendering it practically unusable. Hence, there is a need to recover these lost resources to be able to maintain a proper functioning network.

In the event of a disconnected peer, a *takeover* peer among the neighbors is tasked to recover these abandoned resources while the rest are tasked with recovering the connection. If the neighbor of the leaving peer is not a takeover peer, the peer will begin the process of connection recovery; routing a request message through the network to connect to the assigned takeover peer. The search algorithm used to find the takeover peer is loosely based on `breadth-first` search.

To aid in the recovery process of the network, the concept of a predecessor tree is used along with the network recovery process detailed in [15] is borrowed. The predecessor tree is essentially a binary tree whose primary function is to record



(a) Original Online Setup



(b) Modified Adhoc Setup

Fig. 3. Evaluation Setups

how the zones of the Cartesian space were split. Certain conventions were employed to provide predictability.

IV. Evaluation

A. Setup

We compare the following setups: 1) original CONCLAVE in an online environment; and 2) modified CONCLAVE in an offline ad hoc environment. Five physical devices were used to act as separate peers in the network, an estimation of the usual number of people collaborating on a document based from observation. It should be enough to evaluate the modification done on the peer-to-peer handling side of the CONCLAVE application. All devices run on the Ubuntu Linux distribution OS.

1) Original Online Setup: In this setup, as shown in Fig. 3.a, no modifications were made to the original application and an internet connection is available to all collaborating peers. This setup serves as the baseline for the modifications made in this project. The web hosting service GITPOD was used for deployment.

In the original client, a sharing link is necessary as a means for discovery. We use the web application `Pastebin` which allows the pasting of text and sharing this to other users on the web. This was utilized because it has a relatively shorter link that can be manually copy-pasted. It also minimizes the bandwidth necessary to send the link compared to the usual messaging apps.

2) Modified Ad Hoc Setup: For this setup, as shown in Fig. 3.b, the application is tested with the modifications made to peer handling and peer discovery in an ad hoc network. The ad hoc network is created by means of a mobile phone hotspot (Google Pixel 6A). The hotspot solely serves as a bridge for the collaborating peers and does not have access to the internet.

B. Metrics

Four main metrics were measured in evaluating each configuration: a) *packet delay*, b) *jitter*, c) *data rate*, and d) *peer disconnection resolution time* (PDRT). Wireshark, a packet analysis software, was used to obtain these metrics. Measuring delay, jitter, and data rate can be done in a straightforward manner using Wireshark.

For packet delay, Wireshark can plot the graph of the round-trip-time (RTT) of the packets. From which, the packet delay can be extracted. Packets sent from one peer will have a corresponding timestamp on the source device and a receiving timestamp on the destination device which allows time difference calculation to record packet delay and jitter.

For data rate, Wireshark shows the data rate of the live capture from start to end. By setting time intervals between live captures and setting them to have specific time lengths, we can measure the data rate of each instance. The type of protocol used for transport was considered, namely UDP, and whether the method for measuring the following can be done for both local and ad hoc networks. For TCP, we can easily measure the RTT of a packet because there is a response to every sent packet. For UDP, this reply needs to be forced such that an acknowledgement of the received packet is obtained. The capture file from each peer is required for the measurement of the packet information. A traditional approach to measuring UDP packet information was used [16]. This is a taxing approach if it is to be done to all the data packets. Only five packets from each P2P conversation are randomly chosen to estimate the metrics for each setup, due to time constraints.

PDRT is a newly introduced feature in the modified CONCLAVE application, hence there is no comparison to be made with the original CONCLAVE application. That being said, PDRT is still measured to gauge the performance of the adopted implementation. This is the time it takes for the whole network to reach a stable state when a peer disconnects.

The PDRT is measured through the web console. Print commands were set in place within the code to keep track of the state of each peer in the overlay network. The starting point and end points of a disconnection process is identified manually. Then, the time difference between the two events measure the PDRT of the specific peer in the network. The end of a recovery process within a specific peer is the point at which its state becomes static. Since this is a distributed network, the actual PDRT of the entire network is effectively the peer with the longest PDRT.

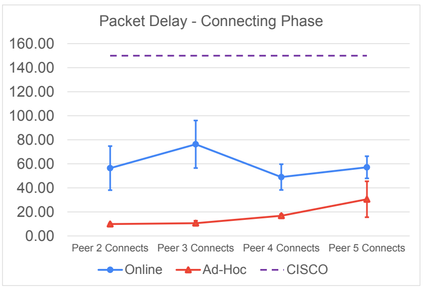
As a benchmark, Cisco's recommendation on the desirable metrics for VoIP implementations was used [17]:

- Maximum one-way latency of 150 ms
- Average one-way jitter under 30 ms
- Loss no greater than 1 percent
- A range of 21 to 320 kbps of priority data rate must be allocated for the conversation.

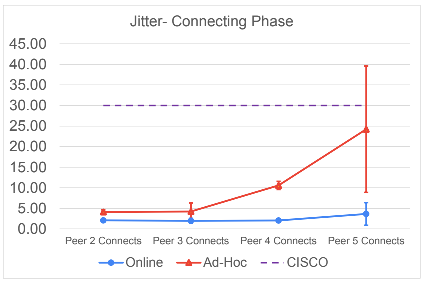
The Cisco metrics might be too stringent since CONCLAVE is not a VoIP application, but a text collaboration application. These recommendations were the closest available recommendations especially for real-time applications. Quality of service standards for text messaging do exist but have lenient requirements that are not applicable for this type of real-time application [18].

C. Evaluation Phases

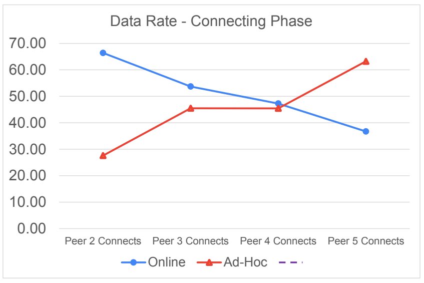
Since the approach for measuring UDP packets is done manually, or without the use of a third party application, some organization in the message exchange methodology is also required. Measurement of the necessary data is split into two phases, the *connecting* phase and the *collaborating* phase, with the intention to segregate packet contents to their respective phase. For the collaborating phase, it is further



(a) Packet Delay



(b) Jitter



(c) Data Rate

Fig. 4. Connecting Phase

subdivided into *idle* and *active* phases. This is to provide insights on the baseline metrics during low traffic scenarios, particularly data rate, as compared to high traffic scenarios. Evaluation was also done on the PDRT of the modified CONCLAVE implementation.

V. Results and Analysis

The measurements for the connecting phase and the collaboration phase are shown in Fig. 4 and Fig. 5 respectively.

1) Connecting Phase: In terms of packet delay, as shown in Fig. 4.a, the ad hoc setup performed well below the maximally defined measurement recommendation by Cisco. The routing path of the packets may affect the packet delay as observed in the difference of the third peer connection behavior in the online vs modified setup.

As shown in Fig. 4.b, the jitter of the ad hoc setup shows a steep rise after more peers are connecting. Compared to the online setup, where there is not much variation at higher peer counts, the online setup provides a better performance.

The ad hoc setup has a steadily increasing data rate (as shown in Fig. 4.c) with each addition of a peer. It can be observed that, with the addition of peer 4, there is no increase in data rate since no extra addition is needed with peers that are not neighbors. In the online setup, the trend is in the opposite direction. The data rate decreases as more peers are included.

2) Collaborating Phase: In the collaborating phase, the entire sample of packets are comprised of the packet exchanges between each peer conversation in the network. In this phase, both setups are now a peer-to-peer network without the data transfer to an online server.

As shown in Fig. 5.a, the ad hoc setup displays a performance at par with the online setup. The online setup seems to have relatively good performance at smaller network sizes. It is expected that, beyond the 5-peer mark, the performance should worsen significantly due to the original CONCLAVE's branching star topology which does not scale well.

For the jitter plots, as shown in Fig. 5.b, the ad hoc setup performed outstandingly well in comparison to the online setup, having relatively more consistent and lower variation in the measurements at all network sizes.

The plots for the data rate shown in Fig. 5.c are as expected. Both setups follow a similar trend, although, it can be observed that the ad hoc setup has an overall higher base data rate compared to the original online setup. This is due to the distributed nature of the modified implementation where additional background process, otherwise handled by online servers and other existing infrastructure in the online setup, are necessary to maintain a distributed network.

TABLE I
Local Network Configuration - PDRT

Case	Peer Disconnect Resolution Time (s)				
	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5
P1 Leaves	-	**8.846**	8.811	0	8.807
P2 Leaves	**31.662**	-	7.511	31.43	25.578
P3 Leaves	12.116	20.99	-	**33.246**	33.127
P4 Leaves	13.256	**31.398**	31.7	-	13.202
P5 Leaves	**6.267**	6.21	6.209	0	-

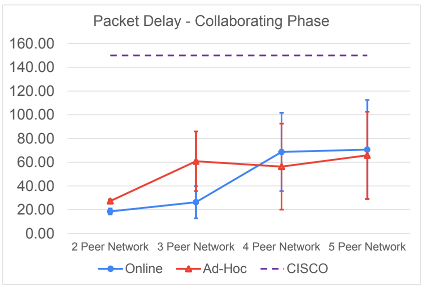
A. Overall Analysis

The quality of service metrics do not fully represent the user experience, as other factors such as processing delays and external influences could also impact the application's performance, which is discussed in this section to help with future development decisions.

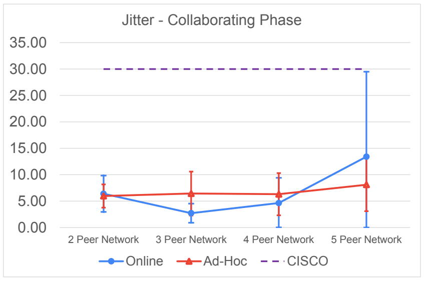
1) Performance of Modified Ad Hoc Setup vs. original Online Setup: The ad hoc setup shows the best performance in the tests in terms of minimal delays and scalability while being in accordance to the established recommended performance range. While it cannot be seen in the result, due to the constraint of a 5-peer-maximum network, it is expected that the modified set-up will out perform the original setup dramatically beyond the 5-peer mark due to the nature of the peer handling used.

In terms of user experience, the ad hoc setup also had the best performance. The online original setup displayed some visible delays, relative to the responsiveness experienced in the ad hoc setup. Although, the difference in the delays are practically insignificant in a normal setting. The online setup metrics is comparable to the ad hoc setup, this difference in user experience is attributed to the difference in peer handling used.

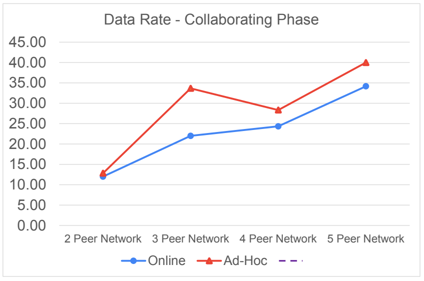
2) Peer Disconnection Resolution Time: The results shown in Table I is heavily dependent on the network structure and position of the leaving peer in that network. The implemented delays in the processing is necessary to ensure organization and ordering in the processes for network stability in distributed networks. However, some optimization in this part is still possible. It should also be noted that, during disconnection resolution, the collaboration of the peers in the network are unimpeded. The important part of the disconnection resolution is the fast recovery process which handles recovery of abandoned resources in the network,



(a) Packet Delay



(b) Jitter



(c) Data Rate

Fig. 5. Collaborating Phase

which happens almost instantaneously after a peer leaves. The remaining connection resolution, which is being measured, is not as crucial. In the event of multiple simultaneous disconnections, a network failure is observed and is outside the scope of the project.

VI. Conclusion and Future Work

In conclusion, offline real-time collaboration was achieved in this project by modifying certain properties of the CONCLAVE web application. This included a new peer discovery system that simplified the whole process by abstracting the sending of invite links through WebSocket messaging. Also, a new peer handling was successfully implemented that presented a peer disconnect resolution process using a version of a Content Addressable Network (CAN). This modified version of CONCLAVE was tested in an offline ad hoc network and was fully functional in this set up. Moreover, the modified ad hoc setup performed marginally better than the online original setup which primarily can be attributed to the implemented peer handling.

Recommendations and Future Work

To improve this web-based implementation, it is recommended that the PeerJS abstractions are replaced by the standard WebRTC API to allow for more precision when modifying the code. This may open potential for optimization of the code. Alteroriginally, the application can be converted and developed into a standalone application where all the necessary services such as the express server can be built into. In a standalone implementation, other means of peer discovery might be needed to support this. Additionally, in doing so, the use of WebRTC may also require some alteroriginal substitution. Not to mention, it may also resolve possible browser related limitations on ad hoc support.

Lastly, the current testing methodology may not have yielded the most precise data to properly estimate the performance of the application. In that, the current data is based on a relatively small sample size. This is primarily due to the time consuming nature of the data extraction process which was not in the favor of the proponents. Perhaps, with the use of more better evaluation tools, to automate this process, it may be favorable to increase the sample size and better estimate the performance of the application. Additionally, further testing the applications with a higher network size for stress-testing and scalability testing could also be beneficial to better understanding the full capacity of the application.

Acknowledgement

The authors acknowledge the Office of the Chancellor of the University of the Philippines Diliman, through the Office of the Vice Chancellor for Research and Development, for funding support through the PhD Incentive Award.

References

[1] N. Savant, E. Olivares, and S.-L. Beatteay, *Conclave*, 2018.
[2] A. Merritt, *Promoting student collaboration in the age of covid-19*, Dec. 2015.
[3] N. L, *Google Docs System design — Part 2— System components explanation micro services arcitecture*. Tech Dummies, Jan. 2019.
[4] S. Weiss, P. Urso, and P. Molli, "Wooki: A P2P wiki-based collaborative writing tool," in *Web Information Systems Engineering – WISE 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 503–512.
[5] N. Lomas, *Collabio lets you co-edit documents without the cloud*, Apr. 2021.
[6] *Building a collaborative text editor with WebRTC and CRDTs*. JavaScriptMN, Feb. 2018.
[7] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun. Surv. Tutor.*, vol. 7, no. 2, pp. 72–93, 2005.
[8] V. Sharma and A. Vij, "Broadcasting methods in mobile ad-hoc networks," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida: IEEE, May 2017.
[9] D. P. I. I. Ismail and M. H. F. Ja'afar, "Mobile ad hoc network overview," in *2007 Asia-Pacific Conference on Applied Electromagnetics*, Melaka, Malaysia: IEEE, Dec. 2007.
[10] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Stabilization, Safety, and Security of Distributed Systems*, X. Défago, F. Petit, and V. Villain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 386–400, ISBN: 978-3-642-24550-3.
[11] R. Linus, *Snapdrop*, 2022.
[12] A. Popescu, D. Ilie, and D. Kouvatsos, *On the Implementation of a Content-Addressable Network*.
[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, Aug. 2001, ISSN: 0146-4833. DOI: 10.1145/964723.383072.
[14] M. Gouda and T. McGuire, "Accelerated heartbeat protocols," in *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*, 1998, pp. 202–209. DOI: 10.1109/ICDCS.1998.679503.
[15] S. Ratnasamy, *A scalable content-addressable network*, Berkeley, CA, USA, 2002.
[16] K. Knochner, *Udp packets' jitter and delay*, Jul. 2012.
[17] Cisco, *Quality of service design overview*, 2020.
[18] J. Babiarz, K. Chan, and F. Baker, "Configuration guidelines for diffserv service classes," Aug. 2006. DOI: https://doi.org/10.17487/rfc4594.