International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 14, Number 5 (2021), pp. 449-455

Implementation of Consensus with TCP/IP

Parshal Chitrakar

Department of Electronic and communication Engineering (ECE)

SRM AP University, Mangalagiri Neerukonda Tadikonda Rd, Mangalagiri, Mandal, Andhra Pradesh 522502, India.

Abstract

Distributed systems need to work with multiple processes and hence, the need for consensus among them is paramount. Large number of such technical processes is implemented on TCP/IP communication framework. TCP is used in conjunction with IP to maintain a connection between the sender and receiver to ensure packet order and to exchange data. The interfaces and protocols needed by the users on the application layer leverage the implementation of consensus to help all the nodes communicate and evolve to a common state. In this work, we use Python3.7 socket programming to demonstrate consensus between two programs running on a network. Further, these programs implement their own version of consensus protocols to achieve a single state. We demonstrate this in action for various times integration steps which can be both matched i.e. same integration time steps on both sides and mismatched, i.e. different time integration steps for both peers. Role of increased round trip delay or RTT has also been explored by making the programs sleep for some predetermined time interval. Insights into implementation of consensus using TCP-IP have been documented.

Keywords: Consensus, TCP/IP, Sockets, Delay, Integration step size.

I. INTRODUCTION

In any connected network, it is important that there are processes in place that can ensure everyone agrees on what information to add and what to discard. These rules or the state is known as consensus and the protocol is known as consensus protocol. They verify transactions and help to keep the network safe. A distributed ledger is spread in the network to verify the transection of the network. The verification is done with the help of consensus protocol like distribution algorithm, Proof-of-Work, Proof-of-stake etc.

Whenever we talk about the TCP/IP we think about the connection but it is way more than that in the sense it can support dynamical processes among automated agents like consensus. Taking advantage of this implementation, we demonstrate a simple consensus algorithm running between two nodes connected using TCP-IP. Our implementation has been programmed to exhibit rich behavior depending on different applications. Before implementing the complex layer on the TCP/IP we first simulated consensus equations in Mat lab for four nodes as shown Fig. 4.Clearly, all four nodes are shown to be approaching a single state with the evolution of time. This consensus behavior of given equations acts as a prototype for implementation of our TCP-IP based peer-message exchange and convergence of their states.

In this work, we will implement the consensus with two communicating nodes, where states are being exchanged between them as they approach each other. There are two ways of achieving this kind of consensus: synchronized and unsynchronized approach. In TCP-IP if congestion is missing, we will witness mostly a synchronized convergence of states among these two participating nodes.

This paper contributes towards following developments.

1. While there is alot of theoretical studies this is one of the unique implementations of consensus on TCP-IP using Python3.7.
2. This work studies the consensus behavior with different integration step sizes Δt in both matched and mismatched modes.
3. Further, we explore the effect of delays on the consensus behavior by implementing sleep command for a predetermined interval.
4. This software is evolvable in the sense of increasing the number of nodes and customizing their behavior by reprogramming consensus equations, integration step sizes, different return trip times, network topologies etc.

In the rest of the paper, we describe related work in section II, implementation workflow in section III, consensus equations in section IV, TCP-IP experimental setup in section V, consensus results in section VI and finally we conclude with insights and future directions in section VII.

II. RELATED WORK

In March 18, 2018 Morsing's blog posted regarding the two nodes with getting into consensus. In which it explains theoretically how TCP is not different from the consensus used in block chain [7]. Naxos: A named data networking consensus protocol, 28-30 June 2018, named data networking lack of the consensus protocol so Naxos adapts the self-learning mechanism to improve the performance. [8].

In April, 2016 M. Stenberg and S.barth published a paper named as "Distributed Node Consensus Protocol"[11]. This

explains how the distributed node consensus protocol can be helpful for the generic state synchronization. This works on the Trickle algorithm and hash trees.

In 2019, Karim Sonbol and Öznur Özkasap published the paper at IEEE conference,"Review on RDMA-Enabled Consensus Protocol". [12] In this paper they explained how our cloud based computing application creates unnecessary data on kernel TCP/IP layers. They developed the Remote Direct Memory Access(RDMA) in order to provide fast communication and that will overcome the overhead copied data. They used the primitives of the RDMA to improve the efficiency of the consensus protocol.

In 2016, On 26th International Conference on Field Programmable Logic and Applications (FPL), David Sidler,Zsolt István and Gustavo Alonso publish their paper on "Low-latency TCP/IP stack for data center applications".[13] Which explains about the reliability, assumption and latency of the TCP/IP also gives an analytical view on TCP/IP

Next we start explaining the components of our work with TCP/IP.

A. TCP/IP

Due to the use of sequence numbers and acknowledgements, the Transmission Control Protocol (TCP) gives us at least some approximation to a reliable link for low congestion and low noise regime anyway. TCP establishes point-point connection: that is, they must both agree on the connection being established. For our purpose, we use TCP-IP to implement, two-party consensus problem or the agreement. As the data is passed down the stack in the network, each layer adds the control information to ensure proper delivery of the packets. There are multiple protocols for sending the data from one node to another node. TCP/IP is the common and user friendly protocol in the present situation. Which consist of five different layers and each layer will add the specific block on the data given by the application layer which are shown in FIg. 2. TCP/IP is implemented using the socket programming in real time where every packet that passes through each layer will help to bind, listen, accept, connect, read, write etc.

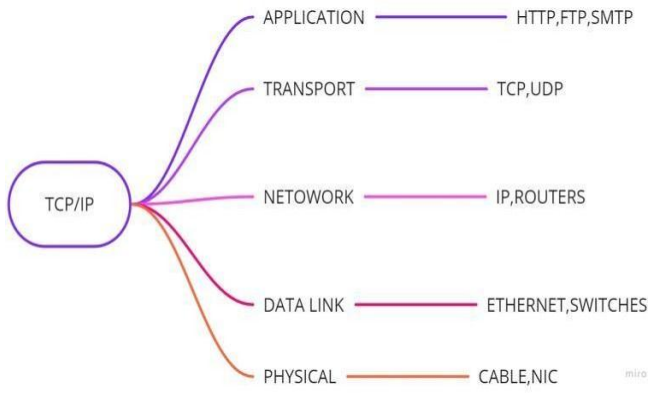

Figure 1: TCP/IP data flow

III. WORKING FLOW

Our main aim is to implement the consensus protocol on the TCP/IP model. Here in our workflow we are going to augment the transport layer by the consensus layer where we are going to implement consensus protocol. There are multiple protocols like distribution, POW, POS etc. which will be working as the consensus protocol in the consensus layer.

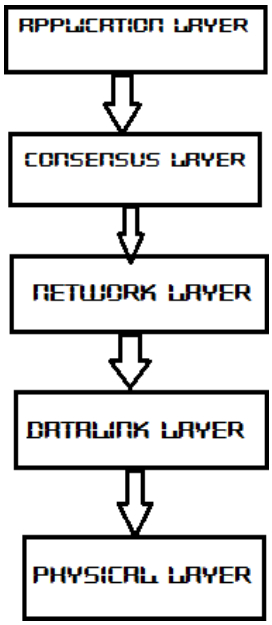

Figure 2: consensus layers

On the application layer, the data is generated on the web browser then the data is sent to the consensus layer, where the data is then encrypted and undergoes the consensus protocol like POW or POS then sent to the network layer where the IP address of sender and receiver is added. After getting the IP of source and destination then the unique MAC address is added on the Data Link layer then the full frame is sent to the wired or wireless network which is done by the physical layer in the form of bits.

As we can see that for sending the data via TCP/IP layers we are going from top to bottom but for the receiver side, the flow will be from bottom to top. If we compare the TCP/IP actual model and this model we can find the difference in transport layer. Here we replace TCP with the Consensus Protocol. In TCP the main purpose is to make the two way handshake between two nodes but in consensus implementation protocol we will try to make each node to be in the same state so that all the nodes won't have to do handshake individually. Here there won't be involvement of any third party so the agreements, connections and the transaction is done within the network members because of achievement of the Consensus.

IV. EQUATION OF CONSENSUS

We start with a simple model of generalized absolute nonlinear flow in consensus over an undirected graph G(V,E)

with degree matrix D and adjacency matrix A [3]. This model can be generalized to a dynamic set-up of connectivity easily.

$$\dot{x_i}(t) = \sum_{j=1, j \in Nbd_i} k_{ij}(f_j(x_j(t)) - f_i(x_i(t)))$$

The above equation is described as absolute nonlinear flow in [9] for ith agent's state evolution equation without feedback delays which can be incurred due to propagation, queuing or processing and can be significant in the case of unavailability of underlying physical layer or during congestion [10]. In this work, a significantly delayed version of eq. 1 is proposed where for ease of presentation and delay symbol book-keeping, information delay has been assumed to 0 << T over all links which can be thought of maximum delays. Nbd_i is a representation of a neighborhood of node i i.e. collection of nodes directly connected to node i.

$$\dot{x_i}(t) = \sum_{j=1, j \in Nbd_i} (f_j(x_j(t)) - f_i(x_i(t))) \equiv -Ł(G)f(x)$$

where Ł(G) := D(G) − A(G) is Laplacian of underlying connecting graph G, D(G) is degree matrix and A(G) is adjacency matrix.

V. EXPERIMENT SETUP

In this paper, for implementing the consensus with TCP/IP we used Python 3.7 as well as matlab. For the four node simulation as shown in Fig. 4, we use matlab and for TCP/IP programming and consensus calculation we used Python Programming language.

For the real time work we use socket programming for creating the TCP/IP connection so that we have to install some libraries like socket, time,numpy, os and for the visualization of data we can install matplotlib which is an inbuilt Matlab library for python. After installing the required libraries we created the two nodes (server and client) python file then we set up the socket programming for both of them. This will give access to the server and client for two way communication. Then the consensus equation is deployed in both server and client so that in every iteration they will send and receive new updated locations until and unless they get into single state or Consensus. The updated location is stored in an array then using the matplotlib we are able to plot the graphs. All the outcomes shown in above results are generated using matplotlib library of python.

A. Flow Chart

To achieve the consensus between two nodes with TCP/IP consists of proper steps for the communication which is shown in Fig. 3. The work flow can be described by following steps:

1. Two nodes, server and client are started or initiated. For us we implemented it with python. So the required packages and libraries are installed for both server and client.
2. After initiation of required libraries and packages we have set up the client and server with the socket. setup consists of socket, bind, listen, accept, connect, read, write and close. These are the building blocks for the setup of our TCP connectivity.
3. When the two nodes are connected or the server and client are connected then the server will send the initial position to client and client will also do the same via TCP/IP.
4. After exchanging the initial position now they began to calculate the consensus equation which is mentioned in the equation section of this paper. In every iteration they will send and receive the calculated consensus position.
5. The loop of calculation of consensus and Rx/Tx goes on until both nodes satisfy a single state which is the achievement of consensus.
6. After getting the consensus both server and client will stop sending the information and get back to the rest step and wait for the initiation.

Next we describe the flowchart for this implementation for ease of understanding.

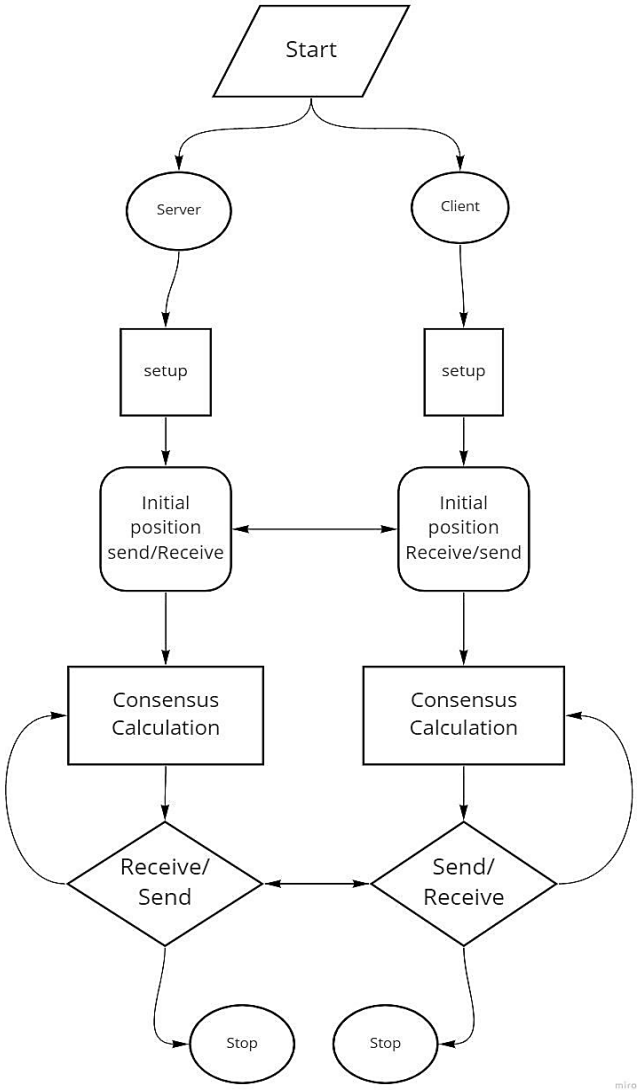

Figure 3: Flow Diagram for two node consensus communication.

Fig.4 shows the development environment implemented in Python 3.7 where all positions are being updated as TCP/IP packets. It makes the idea of consensus far more realistic as compared to just Mathematical equations in an abstract fashion.

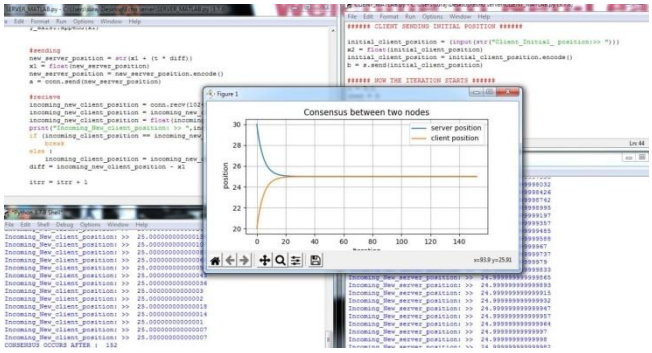

Figure 4: Screenshot of Python3.7 based programming environment to implement the consensus

Equation on TCP/IP. Unlike numerical simulation, here all the position updates are being transferred as TCP/IP packets. It should be considered a step towards realism as compared to abstract mathematical consensus equation

VI. RESULTS

Taking four nodes into consideration in a network, using the MATLAB simulator with the help of the equation mentioned above, can achieve the consensus graph as shown in Fig. 5. In this figure, we can see that the four nodes in the network finally converge to the same state after some time.

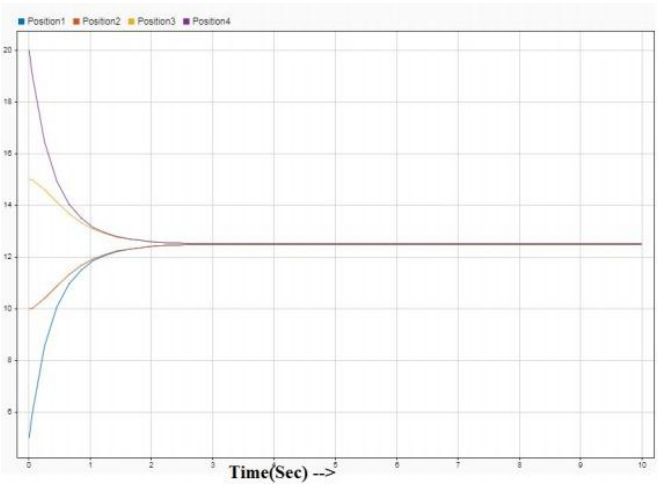

Figure 5: Consensus simulated graph for four nodes

VI.I Consensus Observation

In this section, we present results of TCP/IP implementation in Python 3.7 code with socket programming.. We present results for different integration steps in both matched and mismatched mode as shown in Table-I.

A. Variation on Initial step time(Table 1)

Client		Server	
int. pos.	step time(t1)	int. pos.	step time(t2)
10	0.1	20	0.1
10	0.25	20	0.5
10	0.5	20	0.75
10	0.75	20	0.5
10	0.75	20	0.75

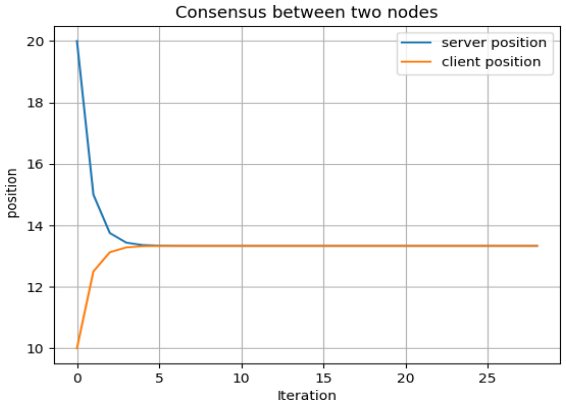

Figure 7: Consensus graph for t1 = 0.25 and t2 = 0.5 from TCP/IP Simulation.

1. t1 = 0.1, t2 = 0.1

In Fig. 6, we showed two nodes converging toward consensus using TCP/IP communication with the server integration step size t1 = 0.1 and client step size t2 = 0.1. we can see the convergence without any fluctuation because of the same step time or matched mode.

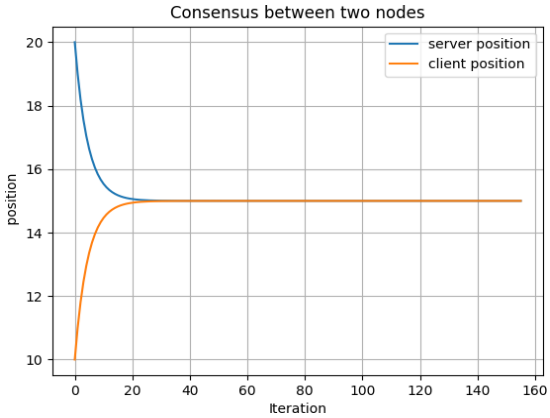

Figure 6: Consensus simulated graph for t1 = 0.1 and t2 = 0.1 from TCP/IP Simulation.

2. t1 = 0.25, t2 = 0.5

In this Fig. 7, we showed two nodes converging toward consensus using TCP/IP communication with the server step size t1 = 0.25 and client step size t2 = 0.5. We observe that the graph of the server and client is getting blocky rather than a smooth graph as shown in Fig. 6.

3. t1 = 0.5, t2 = 0.75

In this Fig. 8, we showed two nodes converging toward consensus using TCP/IP communication with the server step size t1 = 0.5 and client step size t2 = 0.75. We can see the oscillation in the output graph before converging as the variation on the step time increases. More the step time, the bigger the peak of the oscillation.

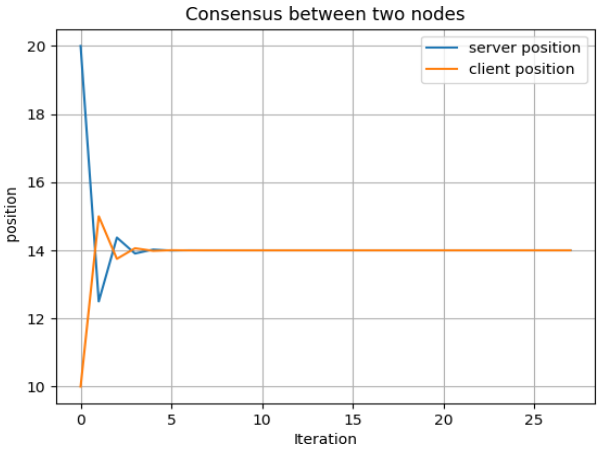

Figure 8: Consensus graph for t1 = 0.5 and t2 = 0.75 from TCP/IP Simulation.

4. t1 = 0.75, t2 = 0.5

In this Fig. 9, we showed two nodes converging toward consensus using TCP/IP communication with the server step size t1 = 0.75 and client step size t2 = 0.5. By interchanging the step time the graph also seems to be interchangeable. (Refer to Fig. 7).

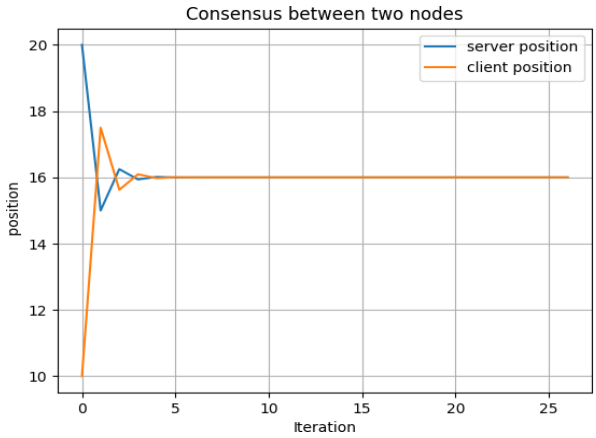

Figure 9: Consensus graph for t1 = 0.75 and t2 = 0.5 From TCP/IP Simulation.

5. t1 = 0.75, t2 = 0.75

In this Fig. 10, we show two nodes converging toward consensus using TCP/IP communication with the server step size t1 = 0.75 and client step size t2 = 0.75. As we can see, increase in step time results in larger oscillation in server and client positions.

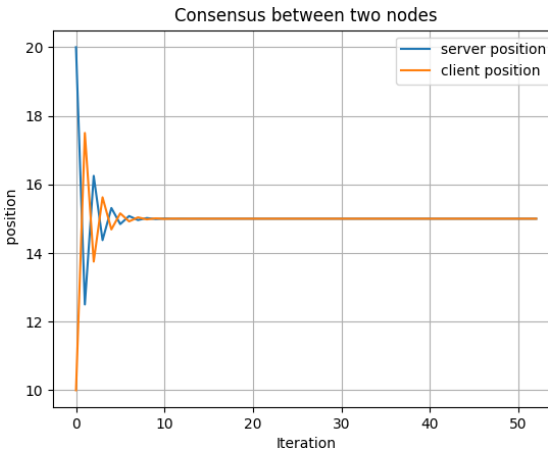

Figure 10: Consensus obtained graph for t1 = 0.75 and t2 = 0.75 from TCP/IP Simulation.

B. Effect of delay in consensus(Table 2)

Client			Server		
int. pos.	step time	delay	int. pos.	step time	delay
10	0.1	0.1	20	0.1	0.1
10	0.1	10	20	0.1	10
10	0.1	50	20	0.1	50
10	0.1	100	20	0.1	100
10	0.1	200	20	0.1	200

1. Delay = 0.1

Now keeping the step size constant, we added delays in both server and client sides. initially the server will send its location via TCP/IP to the client then the client will also do the same. Now both will calculate the consensus then sleep for some time i.e delay is induced, then again send the updated position which can also shown below Fig. 11 with the delay of 0.1.

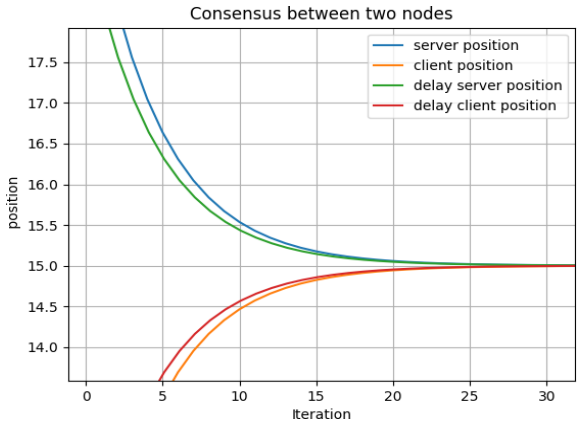

Figure 11: Effect of delay = 0.1

2. Delay = 10

As we can see that the increase in delay, consensus time is also increasing with respect to the applied delay as shown in Fig. 12.

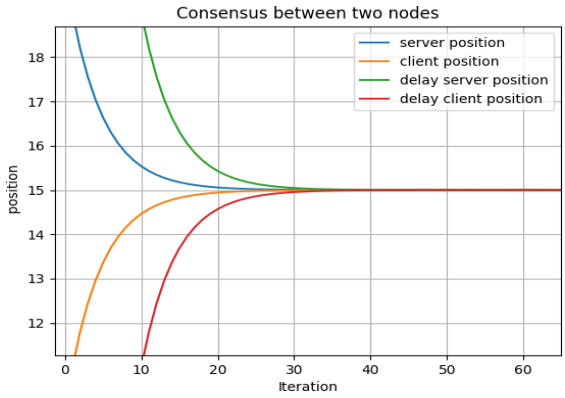

Figure 12: Effect of delay = 10

3. Delay = 50

When we increase the delay further up to 50, we can see the shifted version of the consensus as shown in Fig. 13.

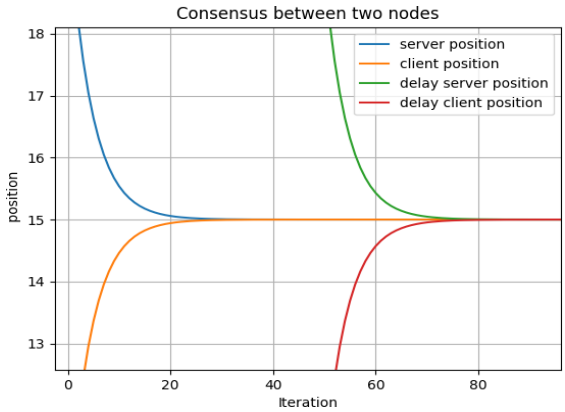

Figure 13: Effect of delay = 50

4. Delay = 100

When we keep the delay of 100 for two nodes then we can see that the t iteration has shifted by t + 100 according to the Fig. 14.

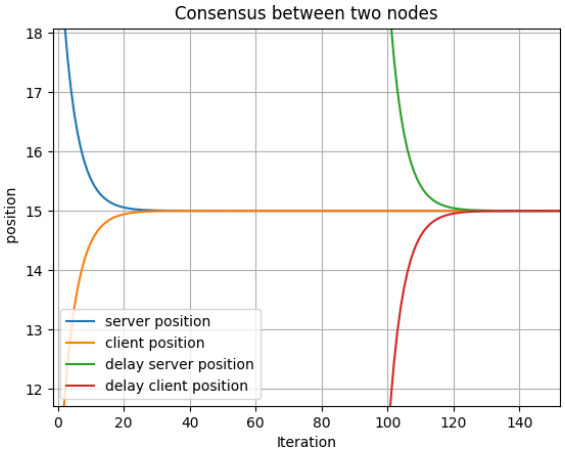

Figure 14: Effect of delay = 100

5. Delay = 200

Fig. 15 shows that the more the delay in sending the position results in more time to reach consensus for two nodes and can be applicable to more than two nodes.

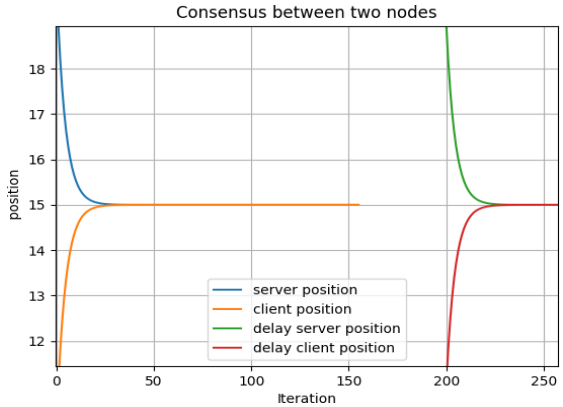

Figure 15: Effect of delay = 200

VII. CONCLUSION AND FUTURE DIRECTIONS

In this paper we proposed a consensus based TCP/IP network with the addition of the consensus layer in the TCP/IP, which will help for any machine/Robot to communicate and can have a mutual understanding. It can also be considered to be the secure way of having a connection between nodes in the sense of group tasks and doing things together. One of major learning from this work is to keep proper sequence of message exchange and updating the positions of the nodes as per consensus equations. If one does not get this right, then it may lead to erroneous results. There are a large number of future directions in this work as it has just started evolving. Some of the straight forward problems are simulations with large numbers of nodes and understanding the scalability. Role of

nodes with complicated consensus dynamics, large delays in message exchanges[3], shortest time convergence, underlying network topologies, optimal resource allocations for high-quality consensus are some of the major unexplored issues which will provide deeper insights into this problem.

REFERENCES

[1] A. Seuret, D. V. Dimarogonas and K. H. Johansson, "Consensus under communication delays," 2008 47th IEEE Conference on Decision and Control, Cancun, 2008, pp. 4922-4927, doi: 10.1109/CDC.2008.4739278.

[2] Srivastava, V., Moehlis, J. and Bullo, F., "On Bifurcations in Nonlinear Consensus Networks," J Nonlinear Sci 21, 875–895 (2011). https://doi.org/10.1007/s00332-011-9103-4.

[3] Priya Ranjan,"Instabilities of Consensus", to appear in Proceedings of CCB,2020.

[4] https://en.wikipedia.org/wiki/Tridiagonal matrix .

[5] https://encyclopediaofmath.org/index.php?title=Jacobi matrix.

[6] https://csustan.csustan.edu/tom/Clustering/GraphLaplacian-tutorial.pdf

[7] https://morsmachine.dk/tcp-consensus

[8] https://ieeexplore.ieee.org/document/8622901

[9] Srivastava, V., Moehlis, J. and Bullo, F., "On Bifurcations in Non-linear Consensus Networks," J Nonlinear Sci 21, 875895 (2011). https://doi.org/10.1007/s00332-011-9103-4.

[10] Ranjan, Priya, Richard J. La, and Eyad H. Abed, "Global stability conditions for rate control with arbitrary communication delays," IEEE/ACM Transactions On Networking 14, no. 1 (2006): 94-107.

[11] M. stenberg, S. Barth,"Distributed Node Consensus Protocol",April 2016

[12] Karim Sonbol,Öznur Özkasap,"Review on RDMA-Enabled Consensus Protocol",2019 International Symposium on Networks, Computers and Communications (ISNCC)

[13] David Sidler,Zsolt István and Gustavo Alonso,"Low-latency TCP/IP stack for data center applications",2016, 26th International Conference on Field Programmable Logic and Applications (FPL)