







# A Hybrid Approach for Synchronizing Clocks in Distributed Systems

Md Shohel Khan , Ratul Sikder , and Muhammad Abdullah Adnan  

Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh  
shohel.expo@gmail.com, ratulsikder121@gmail.com, abduallah.adnan@gmail.com

**Abstract.** The art of synchronizing clocks across a wide area network has got a new dimension when it comes to the reality of achieving the demand for high-accuracy synchronization; even for local or small computing systems. Before implementing any clock synchronization protocol, some important aspects must be considered. For example, communication latency- is it fixed or variable? Does there exist any reference clock in the system or not? In this paper, we have studied the standard and experimental protocols for synchronizing clocks over a geographically distributed network and implemented the features of the Network Time Protocol (NTP) combined with the timing signal of Global Positioning System (GPS) for synchronizing distributed system's clocks more accurately. Our proposed system can achieve higher clock synchronization accuracy compared to the traditional NTP clock synchronization protocol with the help of our designed decentralized GPS-based NTP servers.

**Keywords:** Clock synchronization · Reference clock · Distributed system · Communication latency · NTP · GPS

## 1 Introduction

Clock synchronization in a distributed system can be referred to clock recovery which is achieved by frequent synchronization in serial communication. The mechanism of implementing a clock synchronization system depends on which protocol we are going to implement. Clock synchronization protocols can be distinguished by several significant properties like logical vs physical time synchronization, external vs internal synchronization, a priori vs posteriori synchronization. Another important thing is the performance metrics behind the implementation of clock synchronization in a system. Precision, synchronization error, energy cost, memory requirement, implementation cost, fault tolerance, scalability, etc. are some of the most important performance metrics [14].

The problem of physical clocks is that these are subject to drift and only some precise atomic clocks have a very negligible drift. The difference between any two clocks is called skew. A token ring approach was proposed to solve the problem of synchronizing distributed system's clocks [8]. However, this system is unable to

ensure high precision and scalability. On the other hand, the NTP (Network Time Protocol) is a widely used protocol and it provides microsecond precision whereas the PTP (Precision Time Protocol) provides sub-microsecond precision but this protocol requires PTP-enabled devices. Datacenter Time Protocol (DTP) offers nanosecond precision but it also requires DTP-enabled devices [9].

In this paper, we have proposed a decentralized GPS device-based approach over an NTP network to achieve relative improvements in clock synchronization accuracy compared to the traditional NTP protocol. Our objective is to minimize latency, cost and hardware dependency. According to our proposed mechanism, some low-layered strata computers are equipped with GPS time synchronizers. Relatively small UDP synchronization packets are sent over a secured channel to synchronize with GPS-enabled NTP's (GNTP) neighboring clients' clock. The selection for transforming a member of the network into GNTP solely depends on the stratum number of that member, location of data centers and the accuracy over the cost factor.

For achieving our goal, we have developed an algorithm having the mechanism of comparing the values of the NTP server and the GNTP server's time accuracy to synchronize the clients' clock for higher precision. We first mapped the NTP nodes to find the positions which need to be transformed into GNTP and then we modified the clients and the servers to work with it. The *key contributions* of this work are to modify the NTP servers and the clients to handle the GPS enabled NTP node (GNTP) for neighboring devices' clock synchronization to achieve better accuracy. This includes the NTP network remapping, the appropriate nodes selection, and the synchronization of the neighboring nodes to the GNTP servers. We have also compared the regular NTP synchronization against our modified GNTP clock synchronization.

For testing and evaluating our proposed system, we have set up an appropriate scenario for the experimental analysis. Our system worked fine bearing the common characteristics of other WAN based clock synchronization. Comparing our system with the nationwide NTP clock synchronization servers, we have found a much better accuracy level with the presence of our proposed GNTP node. Both the drift and the RTT become smaller in our GNTP based clock synchronization system.

The rest of the paper is organized in the following sections. In Sect. 2, we briefly describe the existing works as well as the pros and the cons, and challenges of those works on clock synchronization in the distributed system. Section 3 explains our proposed model- a hybrid approach to synchronize distributed systems' clock more precisely than the traditional NTP clock synchronization. In Sect. 4, we have discussed the implementation of our proposed model, the modifications of the NTP server and the client. This section also includes the experimental analysis of the implemented system. Applications of our findings, discussions as well as concluding notes are addressed in Sect. 6.

## 2 Background Study

Clock synchronization plays an important role in establishing a distributed system. For example, if we consider a distributed and live e-auction, the system must know which bidder submitted their bid first for any of the two bidders. In a distributed system, at any moment, the values of all nonfaulty clocks should be nearly equal. If we could find out the difference between two clocks' values, then the result should be negligible. Sometimes knowing the order of the events is enough; this is known as logical time. But, in most of the other cases, the devices need to be synchronized accurately with respect to the physical time.

Every computer contains a hardware clock which operates by counting the oscillation of crystal. To keep track of the current time, the software clock of the computer uses the hardware clock. As the hardware clock varies over time, slowly the time information becomes inaccurate which results in different values of two different clocks at any point in time. This misdemeanor can lose up to 40 ms in every second [6] and the difference between the two clocks is known as their skew. To minimize the clock drifting a token ring approach was proposed where the clocks were synchronized both internally and externally [8].

There are several ways to synchronize the physical clock. If all the computers in a system are synchronized by an external source like UTC signal, then the method is considered as external synchronization. But internal synchronization doesn't require an external source. In this type of synchronization, the clocks in the different computers are synchronized with one another. Internal synchronization is used where time needs not to be necessarily accurate with respect to the precise UTC time [14].

Clock synchronization would be straightforward if the system had bounds on message transmission time. In that case, the lower and the upper bounds on the message transmission time were known. If a process sends a message to another process with its current time  $t$ , then the 2<sup>nd</sup> process sets its clock to

$$t + (\min + \max)/2 \quad (1)$$

and the skew is at most

$$(\max - \min)/2 \quad (2)$$

Where,  $\min$  and  $\max$  are the *lower* and the *upper* bounds of the message transmission time respectively.

Cristian's method [1] is another popular synchronization method and it does not depend on the preset lower and upper bounds of the message transmission time between two synchronizing nodes. Rather when a process  $P_1$  requests current time from another process  $P_2$ , it calculates the Round Trip Time (RTT) of the request. When  $P_1$  receives the current time from  $P_2$ , it sets its clock to

$$t + T_{\text{round}}/2 \quad (3)$$

Reference-Broadcast Synchronization (RBS) synchronizes clocks using physical-layer broadcasts called reference beacons [3]. The receiver nodes use

arrival time for comparing and synchronizing their local clocks and achieved microsecond-level precision to an external time source.

Spanner uses TrueTime clock synchronization protocol for its geographically distributed datacenters. This industry-level high precision protocol guarantees logically incremental timestamps across the datacenters with high constancy. According to Spanner, if any event  $T_2$  starts to commit after another event  $T_1$  finishes committing, then the timestamp of  $T_2$  must be larger than the timestamp for  $T_1$  [2].

Generally, we assume that all the nodes are interconnected in a system, but this is not a permanent scenario. Sometimes the network is more complex; for example, each node knows only one or two of its neighboring nodes. In this case, achieving high precision in clock synchronization is very much challenging. Li et al. (2017), assume each clock as a node in the system [10] and based on different topological structures in a distributed system, they have designed three control algorithms for clock synchronization under three cases- with a reference clock, without a reference clock, and with a fixed communication delay in the networking system.

Madrigal and Tenor (2014) worked on two geographically distributed clock where one clock dwell in a device that is used to acquire sensor data and the other is in a computing system that is used to receive them [5]. They dealt with the problem of estimating the values of relative offset and the clock's rate. They have implemented their proposed method in remote control applications based on remotely transmitted sensor data. Reichman et al. (2015), have worked on a wireless mesh network that is non-centralized and non-hierarchical and has distributed synchronization process [13]. The synchronization process depends on the periodical transportation of messages with the timing data by all the nodes, and the timing messages can be re-transported to other nodes.

Sometimes it is sufficient to be aware of the order of the events. But, if we think of a large distributed system then we need some mechanisms to ensure synchronization in the first place. Park and Kim (2016) have introduced a method which is based on PALS (Physically Asynchronous Logically Synchronous) which is a complexity-reducing architectural pattern for a distributed system which allows the developers to design and implement logically synchronous applications [12].

Yong et al. (2015), have surveyed that the traditional clock synchronization solutions like IEEE1588 and NTP protocols depend on software-based synchronization along with message passing mechanism [16]. As a result, the processing time of software and message transmission delay gain the clock synchronization precision in milliseconds. But that is not an acceptable result where the precision requirement is much higher. To get rid of this situation and make the precision higher, they have introduced a physical clock signal at 1PPS (Pulse Per Second) and that was able to implement external clock synchronization decently. They have used FPGA (Field-Programmable Gate Array) hard logic to track and lock every 1PPS. Using these they have generated synchronization clock (S-Clock) and then the system has finally got higher precision and higher resolution clock in the local crystal clock domain.

Highly reliable and synchronous distributed system is essential for industrial applications. In these systems, the nodes synchronize them with the local clock. Vigner and Breuer (2013) have proved that instead of the local clock, the GPS timing signal can be more effective in terms of synchronization accuracy [15]. The distance between two master clocks can be up to 1000 km. Each master clock updates its time by catching the GPS signal through a GPS receiver. Then the master clocks communicate among themselves to make the system synchronous. We have also implemented a decentralized approach to synchronize the clock in a distributed system.

Clock synchronization protocols like Network Time Protocol and Precision Time Protocol are bounded by the disposition of the packet switching network. Lee et al. (2016) have presented Datacenter Time Protocol (DTP) that omits the data packets transmission and it can achieve nanosecond precision. It uses the physical layer of TCP protocol to implement DTP and it eliminates non-deterministic elements in synchronization protocol [9]. They showed a comparison between different protocols, and we have included our proposed protocol (GNTP) to the comparison table which is illustrated in Table 1.

**Table 1.** Comparison between different existing protocols with our proposed protocol [9]

Protocol	Precision	Scalability	Hardware requirements
NTP	us	Good	None
PTP	sub-us	Good	PTP-enabled devices
GPS	ns	Bad	Timing signal receivers, appropriate cables
DTP	ns	Good	DTP-enabled devices
GNTP	sub-us	Good	Timing signal receivers, cables

The application areas of clock synchronization are numerous. Some of the popular areas are- message delivery, cache consistency, active replication, GPS, GSM, CDMA and so on [7]. For many emerging applications and real-time systems in various domain like automation, smart power grid, etc., having a proper and precise clock synchronization mechanism is a prior condition. Lvesque and Tipper (2016) have surveyed the standard protocols and technologies for clock synchronization in a packet switched network to improve the accuracy when asymmetric delays are present [11]. Under variable channel delays, it is a challenging job but maintaining the cost-effectiveness, precision from microsecond to sub-microsecond is achievable.

### 3 Proposed Method

We propose a decentralized GPS device based approach over an NTP network to achieve improvements in clock synchronization accuracy compared to the traditional NTP infrastructure. We have considered the NTP's hierarchical architecture as a location-based circular layered design which helps to detect the more important geographic areas demanding greater clock synchronization accuracy. These areas could be equipped with a GPS enabled NTP server called GNTP server to achieve higher precision. A proper searching and routing model needs to be developed for implementing the GNTP servers. Our proposed method can be divided into three parts according to its relative domains: (i) NTP hierarchy conversion, (ii) GNTP selection, and (iii) synchronize nearby clocks with GNTP.

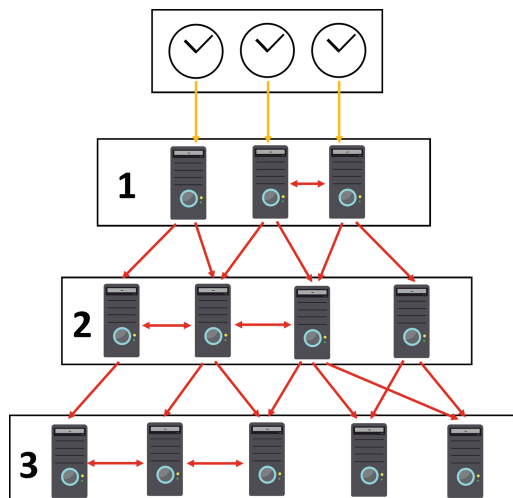
#### 3.1 NTP Hierarchy Conversion

Network Time Protocol is the most used protocol for clock synchronization over a wide area network. NTP protocol generally delivers millisecond-level accuracy over WAN and microsecond-level accuracy over LAN. NTP uses a hierarchical structure and the time source is semi-layered. NTP creates a tree hierarchy and synchronizes all its node with each node's upper stratum layer. The top of the tree is called stratum-0; normally contains an atomic clock, GPS receiver, and other necessary circuits for achieving and maintaining the precise time. Stratum-0 is the time server for stratum-1 computers. Stratum-1 computers are the time servers for stratum-2 computers and so on. It is a server-client based approach where each node except stratum-0, acts as both server and client.

If we assume stratum-0 as the center of a circular NTP network and higher strata nodes as corresponding layers from the center then the overall network forms a circular structure. Stratum-1 devices form the perimeter of the first circular diagram; we can call the stratum-1 perimeter as layer 1 for the corresponding circle. In a similar fashion, we can imagine layer 2 consists of stratum-2 devices, layer 3 consists of stratum-3 devices and so on. The geographical location is considered forming these layers. Figure 2 illustrates this circular NTP network structure. The chronological, directional and geographical properties must be considered forming these layers. This means the circular layer shown in the Fig. 2 is not fully circular. Depending on the geographical coordinates, it becomes a circular type structure where physical locations are the elements of the circle. This circle with the circular layers is the reasoning building blocks of our proposed method.

#### 3.2 GNTP Selection

GNTP is a new terminology used in this paper. It means GPS enabled NTP node. Any node in the NTP network having a GPS time receiver is considered as GNTP time server and thus we will find some additional accurate synchronizer for the network. As GPS time receiver has microsecond level precision, these GNTP servers also have this property (Fig. 1).

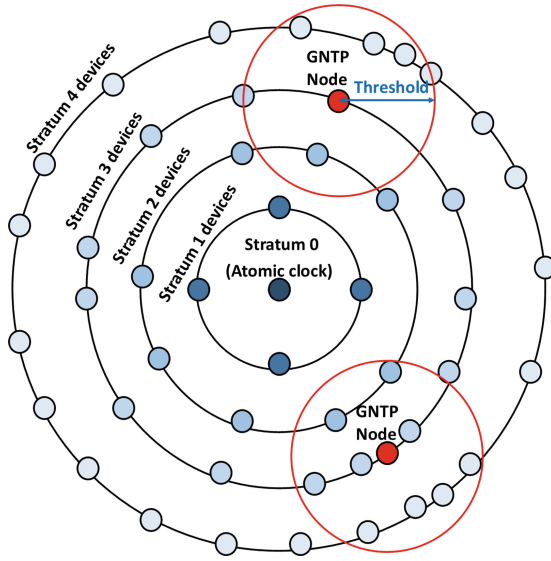


**Fig. 1.** A diagram showing the relationship between the various level of NTP hierarchy. The blue numbers are the stratum numbers; yellow arrows show direct connections, such as RS-232, while red arrows show networked connections [4]. (Color figure online)

There could possibly be some argument against this circular network visualization. The argument of considering a GNTP server as a stratum-0 device and imagining a new NTP network will simply be rejected because the GPS signal cannot be guaranteed to be available always because of the natural conditions. So, a GNTP node needs to synchronize its clock with its upper as well as the same layer devices rarely. Moreover, our proposed GNTP time server design does not involve an atomic clock. So, it can not be considered as a stratum-0 node.

The probability of generating more error to the clock synchronization accuracy on an NTP network is increased with respect to the increasing stratum number of the NTP network's nodes. This is due to the higher packet transmission delay, more middleman devices, long distance, etc. It means that the lower layers of the NTP tree having a high stratum number is prone to more clock synchronization error. NTP nodes can also use the nearby same stratum's nodes to synchronize its clock. It helps the nodes to fix some types of synchronization error but there is still some inaccuracy. By converting some nodes of the NTP network into GNTP, we can achieve more accuracy in the nearby nodes and the corresponding child tree. Here, child tree means the lower layered nodes which are connected to the GNTP node.

Selecting a node in the NTP network for the GNTP conversion is not a straightforward calculation. Whether a node will be selected for converting into GNTP or not, is solely depends on the cost to accuracy factor, the importance of a particular geographic area with respect to the expected accuracy level, etc.



**Fig. 2.** A diagram showing the circular mapping of NTP hierarchy. The GNTP server can be installed on anywhere according to the demand.

### 3.3 Synchronize Nearby Clocks with GNTP

So far, we have proposed and discussed various factors of GNTP time server’s design and goals. Our proposed GNTP is almost accurate regarding the atomic clock. Using these GNTP time servers to synchronize its nearby nodes, especially the nodes demanding more accurate time and the important systems which require more accuracy than traditional NTP synchronization, will improve the accuracy of the corresponding nodes and the accuracy of the child NTP nodes.

Now, the synchronization between our proposed server and client node is also an important factor in achieving higher precision. From our designing perspective and real networking system, this synchronization must be done over a wide area network such as the internet. For clock synchronization over the internet, NTP’s original routing and transporting mechanism is well known and best suited for the purpose. So our GNTP server and client synchronization can be accomplished using the regular NTP protocol; only the timing information of the GNTP servers need to be fetched from the GPS receiver which is highly available. But a proper redirection and NTP server selection rules have to be enforced to each NTP client. The information about the GNTP servers and the relative distance information have to be kept by all the NTP clients. The next section will describe more about the implementation of the GNTP servers and the client nodes.



## 4 Implementation Details

Implementation of our proposed method in a real scenario needs to meet some challenges. The first challenge is transforming the right NTP nodes into GPS enabled NTP nodes or implementing new GNTP servers into the right position in the network. Eventually, it is the result of good NTP nodes mapping of the overall network and the synchronization demand. This includes re-configuring the existing servers and the clients, setting up the new rules and formations, etc. To achieve flexibility, it is possible to implement our proposed method in the software level only without the requirement of extra hardware except for GNTP servers. An extra server needs to be deployed as well for storing the GNTP servers' location which will in turns provide the nearest GNTP server(s) address to the requesting clients.

### 4.1 Implementing GNTP Info Servers

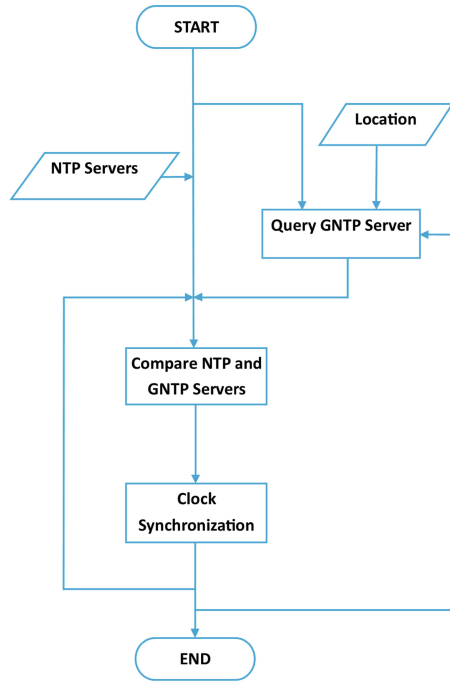
In our proposed method, the first challenge is to provide the nearest GNTP server's address to the clients. To keep the network simple and to achieve greater scalability, we have developed a global GNTP info server to provide the nearest GNTP server's address to the requesting clients.

A GNTP server is a global server which can be accessed by the clients on the network. This GNTP server stores the location information of every GPS enabled nodes in the corresponding network. Client node needs to send the request to the GNTP info server with its own location information to the GNTP info server in order to get the nearest GNTP server's address. The client location needs not to be highly accurate. The location retrieved from the client's very local router can be used for this purpose. In such type of request, our GNTP info server calculates the distance between client and GNTP servers and returns the IP address of the nearest GNTP server if any considering the distance threshold limit which can be configured according to the choice.

The other challenge regarding GNTP info server implementation is how each node of an NTP network knows the existence of a global GNTP info server address. This problem is resolved by a top to bottom approach for passing the GNTP info server's address from stratum-0 to stratum1, from stratum-1 to stratum-2, from stratum-3 to stratum-4 and so on. So, every node configured properly can know the address of the GNTP info server. On the other hand, to ensure the high availability of the GNTP info server, multiple distributed backup servers can be imposed.

### 4.2 NTP Nodes Mapping

In our proposed method, we have meant by *NTP nodes mapping* how we can achieve a circular mapping of NTP nodes, described in the previous section. Each NTP node maintains a file which will contain at least one or a few global servers (GNTP servers) of the corresponding NTP network that contains the address of the GNTP node. The principal job of this server is to provide the nearest GNTP



**Fig. 3.** Data flow diagram of the redesigned NTP client

server address to each NTP server as well as the client based on distance criteria. As GNTP is a static node and its address changes occasionally. New GNTP node is added infrequently as well. So, a typical NTP node will check for its nearby GNTP node in an infrequent fashion i.e. randomly two times per week. This will add very low pressure on those global servers (GNTP servers). Upon a successful GNTP discovery, its address will be locally stored by the corresponding node.

For the nearest GNTP calculation and mapping, each node must have its own exact or approximate geographical location which will be sent to the GNTP servers upon every GNTP discovery request. The GNTP servers must know its' location and update its' own address, and location to the global servers.

### 4.3 Modification to the NTP Servers and the Clients

We used custom UDP packets for our system's clock synchronization. The reference implementation of NTP protocol in the Java programming language is reused and modified according to our demand and structure. We first made our server program to work with GPS device, which delivered us the precise time information. We used this time information throughout our NTP server's system-wide clock. This time is also transmitted to the clients' machine according to the rules of NTP.

**Result:** Synchronize the client’s clock  
**Input :** NTP Servers, Current Location, GNTP Info Server  
**while** *Repeats in week* **do**  
    |  $GNTPServers \leftarrow RequestGNTP(CurrentLocation);$   
    | **while** *Repeats at synchronization frequency* **do**  
    | |  $BestTimeServer \leftarrow Compare(NTPServers, GNTPServers);$   
    | | Synchronize clock with *BestTimeServer*;  
    | **end**  
**end**

**Algorithm 1.** Client Sync

On the other hand, our new modified NTP client checks for the nearest GNTP nodes at the beginning of its lifespan and it will continue checking for it twice per week at a random time. If a nearby GNTP client is found then it saves its address locally. When it comes to synchronization for the NTP client, it will check and compare the minimum delay among other NTP servers and GNTP servers. The server with the minimum delay is chosen and our NTP client synchronizes its clock with that.

We have developed a global server for storing the information about GNTP nodes. This global server’s address is known by all nodes in an NTP network and this small information is passed in the top-to-bottom approach throughout the network. Our client request for GNTP nodes with its own location and in return it gets the closest GNTP server’s information along with the distance between them. Depending on the distance threshold, defined by the system administrator, the client then decides to consider it or to go with the normal procedure of traditional NTP synchronization. We have summarized the concept through a data flow diagram in the Fig. 3.

**Result:** Return the best available time server  
**Input :** NTP Servers, GNTP Servers  
**Output:** Best Time Server  
 $ServerAccuracyList \leftarrow NULL;$   
**foreach** *NTP*  $\leftarrow NTPServers$  **do**  
    |  $ServerAccuracyList \leftarrow GetAccuracy(NTP);$   
**end**  
**foreach** *GNTP*  $\leftarrow GNTPServers$  **do**  
    |  $ServerAccuracyList \leftarrow GetAccuracy(GNTP);$   
**end**  
 $SortAscending(ServerAccuracyList);$   
**RETURN**  $ServerAccuracyList[FirstElement]$

**Algorithm 2.** Compare Time Servers

The concept shown in the Fig. 3 can be properly summarized in the Algorithms 1 and 2. Algorithm 1 shows how to synchronize the client's clock by finding the best available time servers. The best time server is found by comparing the NTP and the GNTP servers. The first loop repeats the procedure once or twice in a week: meaning that the discovery process of the nearby GNTP servers needs to perform merely. On the other hand, the second loop needs to repeat itself according to the required synchronization frequency. This generally varies from 5 to 10s.

The Algorithm 2 shows the normal comparison procedure of the available time servers. This comparison depends on the NTP's *intersection algorithm* which is responsible to find the accuracy of any time server based on NTP protocol. After finding the accuracy of the available NTP and GNTP time servers, the best time server is returned to the Algorithm 1 and then the local system get synchronized its clock with the best time server.

## 5 Experimentation

We have set up an appropriate scenario for our experimentation. Setting up a full geographically distributed NTP network is challenging. Our proposed system's accuracy largely depends upon the availability of nearby GNTP nodes. Considering the situation, we have designed a similar structure for our experimentation. As expected, our experimental results show the basic characteristics of the clock synchronization system over a wide area network. Comparison of our work to the NTP, setting up the servers and the clients, etc are discussed below.

### 5.1 Setting up Servers and Clients

We have run our modified NTP server and client on Linux virtual server as well as a dedicated server. We used VirtualBox 5.2.18 as the virtualization platform. We used LinuxMint 19, a popular version of Linux distribution on our virtual machines and dedicated server for our testing. Our java programmes were run on OpenJDK 11 SDK. We have used two different networks for servers and clients to perform our experiments.

### 5.2 Experimental Results

From the resultant data of our proposed model, we have discussed four important graphs below. The performance and comparison are depicted with respect to different factors.

**Offset vs Drift.** We ran our designed clients and GNTP servers in different machines. The distance between the client and the GNTP server was about six kilometers. A dedicated workstation was used for time server with static

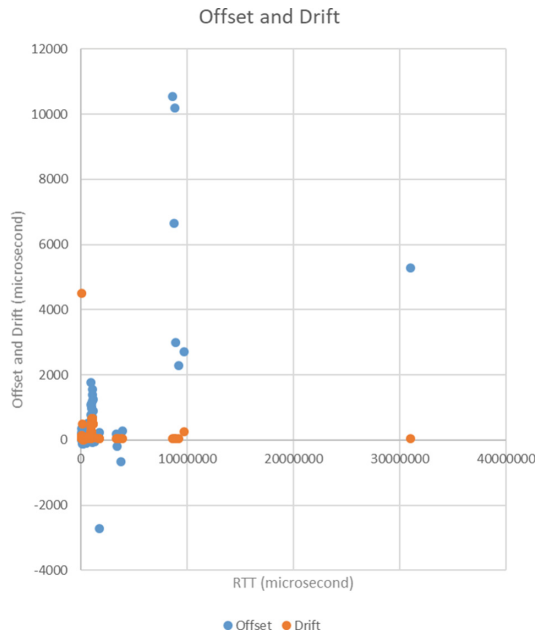


Fig. 4. A resulting graph showing offset and drift against RRT

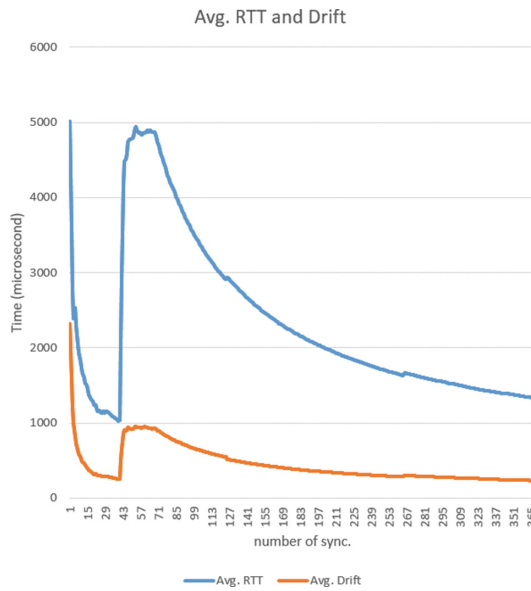


Fig. 5. Average RRT and drift- indicating connection stability.

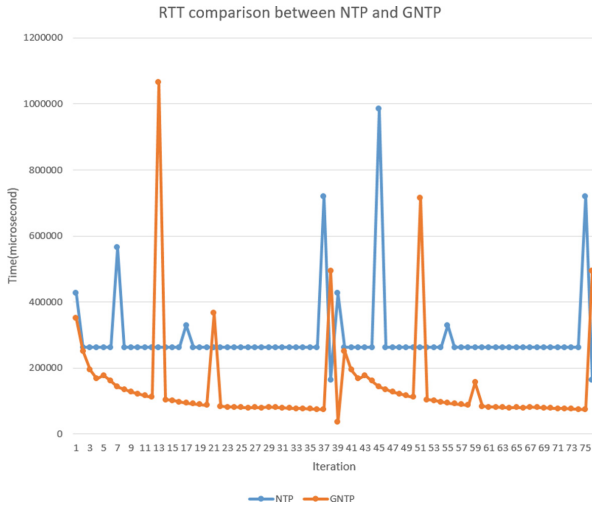


Fig. 6. Round trip time comparison between NTP and GNTP

IP address. We have tested our system’s performance as well as have compared the results with a national NTP server. As the distance between the NTP server and client is greater, the round trip time is also higher.

We have drawn a resulting graph which represents offset and drift with respect to the RTT. The resultant graph in the Fig. 4 shows the non-deterministic nature of the error in clock synchronization regarding the higher round-trip time. The higher the RTT results, the more the probability of synchronization error. From the graph, we see the offset is getting higher (both in positive and negative axis) with the RTT. On the other hand, the drift is more consistent regarding RTT.

**Average Round Trip Time vs Average Drift.** Figure 5 shows the average round trip time and average drift after each synchronization. In the context, synchronization frequency was 10 s. The graph shows a consistent improvement on average RTT as well as average drift. A sudden peak from about 40th synchronization occurred for connection instability between the server and client. This type of connection instability results in increased RTT which directly increases synchronization error.

**Round Trip Time Comparison Between NTP and GNTP.** Figure 6 shows the round trip time comparison between a national NTP server (0.asia.pool.ntp.org) and our implemented GNTP server. The graph shows that the GNTP synchronization takes much lower RTT than the NTP. Lower distance results in lower RTT and greater synchronization accuracy as discussed earlier. The spikes in both the NTP and the GNTP’s curves are result from connection instability, server’s response time, etc. and it is a normal phenomenon in any kind of WAN-based communication.

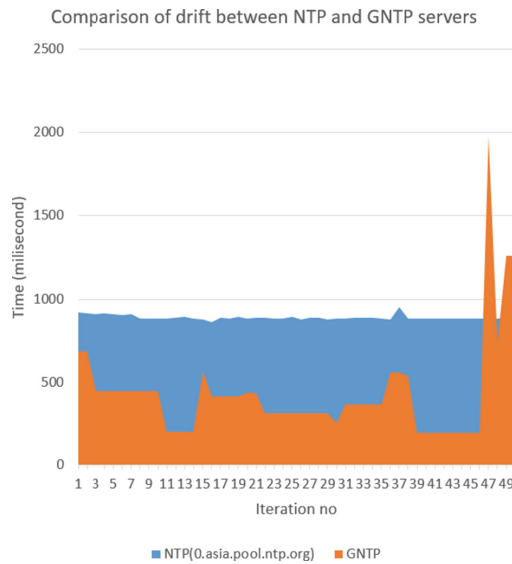


Fig. 7. Comparison of drift between NTP and GNTP servers

**Comparison of Drift Between NTP vs GNTP Servers.** As mentioned earlier, higher RTT is one of the main reasons for error in clock synchronization. The precision gets higher with respect to higher RTT as we have implemented a GNTP server closer to our client, the time inconsistency gets lower. As a result, the synchronization error has minimized and this improves the precision. Figure 7 shows the comparison of drift between a national NTP server (0.asia.pool.ntp.org) and our GNTP server.

## 6 Conclusion

We have implemented a hybrid method for synchronizing clocks in a distributed system which achieved better synchronization accuracy than the traditional NTP synchronization. Installing GNTP servers in more important regions according to the circular NTP hierarchy mapping with respect to the precision demand can decrease the synchronization error- that is a combined result of the lower RTT and the GNTP servers' higher accuracy. In a complex network, it is not always true that the observed nodes are interconnected; perhaps a node may only know its neighboring nodes. A global GNTP info server solves this problem in our work. Our proposed model is implemented and tested in some small-scaled geographically distributed virtual machines. Implementing and testing the system in a geographically large-scaled distributed real machines would led us to examine the existing and probably new factors on this domain more preciously. In the future, we have plans to extend this work in a large-scaled scenario and fine-tune the various threshold values according to the synchronization demand. Our

work could be helpful where better clock synchronization accuracy is needed than the traditional NTP clock synchronization. Small to medium scaled distributed systems, embedded systems, peer to peer networking, as well as every personal computer, can get benefited from our proposed system as clock accuracy is more or less vital in every computing device.

## References

1. Cristian's algorithm - wikipedia (2018). Accessed 24 Sept 2018
2. Brewer, E.: Spanner, truetime and the cap theorem. Google, Technical report (2017)
3. Elson, J., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Oper. Syst. Rev.* **36**(SI), 147–163 (2002)
4. Esham, B.D.: File:network time protocol servers and clients.svg - wikimedia commons, September 2007. Accessed 24 Sept 2018
5. Fernández-Madrugal, J., Martínez-Tenor, A.: Two-clocks synchronization for networked sensors. In: *SENSORS*, 2014, pp. 2022–2025. IEEE, November 2014. <https://doi.org/10.1109/ICSENS.2014.6985431>
6. Ganeriwal, S., Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks. In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 138–149. ACM (2003)
7. Jerzak, Z.: Clock synchronization in distributed systems, September 2009. <https://goo.gl/YvpQDv>
8. Latha, C., Shashidhara, H.: Clock synchronization in distributed systems. In: *2010 International Conference on Industrial and Information Systems (ICIIS)*, pp. 475–480. IEEE (2010)
9. Lee, K.S., Wang, H., Shrivastav, V., Weatherspoon, H.: Globally synchronized time via datacenter networks. In: *Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM 2016*, pp. 454–467. ACM, New York (2016). <https://doi.org/10.1145/2934872.2934885>, <http://doi.acm.org/10.1145/2934872.2934885>
10. Li, G., Niu, M.j., Chai, Y.s., Chen, X., Ren, Y.q.: A novel method of clock synchronization in distributed systems. *Chin. Astron. Astrophys.* **41**, 263–281 (2017)
11. Lévesque, M., Tipper, D.: A survey of clock synchronization over packet-switched networks. *IEEE Commun. Surv. Tutor.* **18**(4), 2926–2947 (2016, Fourthquarter). <https://doi.org/10.1109/COMST.2016.2590438>
12. Park, J., Kim, T.: A method of logically time synchronization for safety-critical distributed system. In: *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 356–359. IEEE (2016)
13. Reichman, A., Priesler, M., Wayer, S.: Distributed network synchronization. In: *2015 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS)*, pp. 1–5. IEEE (2015)
14. Rollins, S.: Time synchronization (2015). Accessed 24 Sept 2018
15. Vigner, V., Breuer, J.: Precise synchronization in large distributed systems. In: *2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, vol. 1, pp. 226–230. IEEE (2013)
16. Yong, C., Hao, W., Xiaofeng, T., Wenbo, W.: Clock synchronization technology based on FPGA. In: *2015 IEEE International Conference on Communication Software and Networks (ICCSN)*, pp. 43–46. IEEE (2015)