Chapter IV

Results and Discussion

In this chapter, simulation results are presented to measure the performance of the proposed algorithms in terms of throughput and average delay in local and global networks by taking the Bernoulli (random) and Poisson traffic distribution models into consideration. Further how the overall latency can be minimized by distributing the data among the servers have been shown in the results and explained. Also, estimation of the network propagation delay has been given to see its effect on overall delay. Finally, a comparative performance of the proposed and the existing PDDRA scheme has been evaluated and explained.

4.1 Performance Evaluation of the Proposed Algorithm

Here, the performance of proposed algorithm is presented by considering different cases. The cases are built by having different assumptions about the request generating nodes, servers buffering capacity and networks and traffic models. Bernoulli traffic model is considered in the results from case I to case V which is by default random model. In case VI, Poisson traffic arrivals of requests are considered.

Case I: It is assumed that at the local network ‘No’ request can be full-filled and therefore all the generated requests will be transferred to the global network. Number of request generating nodes (N) and servers (S) are 4 with varying Buffering Capacity (B) =2, 4, 8 and 16.
In figure 4.1, the throughput vs. average load is plotted with varying size of buffer capacity. While considering that the request generating nodes are four and server nodes are also four. The first clear inference is that, as the buffer capacity increases the throughput also increases. As for $B=2$ at the load of 1, the throughput is 0.88 which increases to 0.98 for $B=16$. It is evident from the figure, to get at least 93% throughput, there is a need for a buffering capacity of 4 requests. It is clear from the figure that even at the higher load > 0.8, throughput up to 98% is achieved.

![Throughput vs Load](image)

**Figure 4.1. Throughputs vs. load for various values of B while considering N=4 and S=4.**

In figure 4.2, the average delay vs. average load is plotted with varying size of buffer capacity. While considering that the request generating nodes are four and server nodes are also four. It is observed that, as the buffer capacity increases the average delay also increases. As for $B=2$ at the load of 1, the average delay is 1 slot, which increases to 10 slots for $B=16$. As expected as buffer space increases, the average delay also increases.
Figure 4.2. Average delay vs. load for various value of B while N=4 and S=4.

It is also clear from the figure, that below the load of 0.5, the average delay is nearly zero, and thereafter it rises exponentially. This is also very obvious that as the load increases more number of requests arrive and to sustain the throughput buffer space has to be increased and thus the average delay also increases. It is evident that if more number of requests are buffered, then throughput improves, and thus due to large numbers of buffering of requests the average delay also increases.

Case II: It is assumed that at the local network some requests can be full-filled and therefore the generated requests will be shared between the local and global networks.

In figure 4.3, throughput vs. load is plotted while; ‘a’ denotes the percentage of traffic that is served globally for example a=80 denotes that 20% of the total generated requests are served locally and 80% of the generated traffic is served globally. It is clear from the figure; if 80% of the total traffic is served globally then 100% throughput is possible. However, as traffic crosses 80% limit the throughput, decreases and attains a value of 96%. Comparing figure 4.1 and 4.3, it is very clear that if load is shared between the global and local networks, the overall throughput increases.
For the similar set of values as in figure 4.3, the load vs. average delay is plotted in figure 4.4. It can be seen from the figure, that when only 20% of the total traffic is served globally then the average delay is zero. For 80% of the traffic the average delay is zero, below load 0.6. Above the load 0.6, the average delay increases and attains a value 1 slot at the load 1. However, if all the traffic is served globally the delay is significant and at the load 1 it attains a value of 5 slots.

Figure 4.3. Throughput vs. load for $B=8$, $N=4$, $S=4$ and varying the values of ‘$a$’.

It must be remembered that, if the same set of values are assumed for the local network and global network, then the queuing structure will remain the same for local and global network. In view of this, considering figure 4.3, then results for $a=20\%$ is for local database and $a=80\%$ are for global database. The total throughput at the load of 0.9 or below is

$$T = \frac{20}{100} T_{av}^L + \frac{80}{100} T_{av}^G = 0.2 \times 1 + 0.8 \times 1 = 1.0$$

However, the total delay will be evaluated as

$$D = 0 + 0.5 = 0.5 \text{ slot}.$$
\[
T = \frac{20}{100} \times 1 + \frac{80}{100} \times 0.998 = 0.9984
\]

The total delay will be evaluated as

\[
D = 0 + 1.0 = 1.0 \text{ slot}
\]

Figure 4.4. Average delay vs. load for B= 8, N=4, S=4 and varying the values of ‘a’.

Considering the case, when 80% of the requests are served locally then only 20% of the requests will be served globally. Even in such a case the throughput will remain same as in the above case. However, the average delay at the local network will be of 1 slot and delay at the global network will be nearly zero. \(D = 1.0 + 0 = 1.0\) slot. This clearly suggests that it may possible that sometimes data may be available locally but the access time of the local network may be higher than the global network. Hence, to send a simultaneous request to both local and global network is necessary to minimize the latency.

In figure 4.5, throughput vs. load is plotted while varying the traffic served by global servers. It is clear from figure 4.3 that if 80% or less of the total traffic is served globally then 100% throughput is possible. Hence for 60% and 40% of the traffic, throughput is 100%.
Figure 4.5. Throughput vs. load for $B=8$, $N=4$, $S=4$, while ‘a’=60% and ‘a’=40%.

For the similar value as in figure 4.5, the average delay values are plotted in figure 4.6. It can be seen from the figure, that when only 60% of the total traffic is served globally then the average delay is 0.22 slots even at the load 0.8. For 40% of the traffic the average delay is 0.04 slots at load 0.8.

Figure 4.6. Delay vs. load for $B=8$, $N=4$, $S=4$, while ‘a’=60% and ‘a’=40%.

Case III: In this case, the assumption that at the local network some requests can be full-filled and therefore all the generated requests will be shared between the local and global networks is retained. But here, it is further assumed that server requests cannot be buffered at the server.
In figure 4.7, throughput vs. load with request generating nodes being ‘4’ while considering storage capacity (B=0) is nil, is plotted. It is seen from the figure, as the buffer deceases, the throughput also decreases.

![Figure 4.7. Throughput vs. load with N=S=4 while B=0.](image)

Comparing this figure with figure 4.1, it is clearly seen that on the overall throughput, buffering has deep impact. As in figure, with buffering capacity nil and with a=100%, the throughput value is below 0.68. In this case, there is no need to calculate average delay as buffering capacity is zero (figure 4.8); the average delay will be zero.

![Figure 4.8. Average delay vs. load with N=S=4 while B=0.](image)
It can be inferred that if request generating nodes are four, then if 20% or more requests are served locally then the buffer space of ‘4’ will be sufficient while the average delay is negligible. As the mean waiting time for local server is nearly zero if load is less than 60%, hence the total waiting time would be equal to the global server waiting time.

Case IV: It is assumed that the numbers of servers that can serve the requests are either 1 or 4. In addition, the following are assumed

(i) At local network no request can be full-filled

This assumption is considered to analyze the effect of number of servers on the parameters throughput and average delay.

![Figure 4.9. Throughput vs. load for B=4, N=4, S=1, S=4 and ‘a’=100%.](image)

In figure 4.9, the number of requests generating nodes are assumed to be 4 while assuming that the number of server are $S=4$ and $S=1$ with $a=100\%$ and buffering capacity of servers are of 4 requests. It is seen from the figure that up to load 0.8, the throughput is nearly one in case of $S=4$, while with $S=1$ it is continuously decreasing.
and it attains a value of 0.5 at the load of 0.5 which is very less. As the buffer of only 4 requests is allowed with $S=1$ the average delay reaches to 4 slots even at load of 0.9, while with $S=4$ it reaches to a value of 2.5 slots at load of 1.0 (figure 4.10).

![Figure 4.10. Average delay vs. load for B=4, N=4, S=1, S=4 and ‘a’=100%](image)

(ii) Some of the requests can be full-filled locally

In this section an analysis is given to see how throughput and average delay get affected when number of servers varies and the traffic is shared between the local and global networks.

In figure 4.11, the number of servers is assumed to be 1 and 4 while assuming that the request generating nodes are four with buffering capacity of four requests. In the simulation, it is considered that the global load is 80%. It is clear from the figure that with $S=1$ and $a=80\%$ the throughput is very less and at the higher load it is nearly zero, while delay is maximum of 4 slots (figure 4.12). As the number of servers increases or load on server decreases the throughput increases and average delay also decreases.
Comparing the case of $S=1$ and $S=4$ with $a=80\%$, the delay in case of $S=1$ is 4 slots while for $S=4$ it is of 1 slot.

Figure 4.11. Throughput vs. load for $B=4$, $N=4$, $S=1$, $S=4$, and ‘$a’=80\%$, ‘$a’=20\%$.

Figure 4.12. Delay vs. load for $B=4$, $N=4$, $S=1$, $S=4$, and ‘$a’=80\%$, ‘$a’=20\%$.

In figure 4.13, results are generated with varying number of servers. The numbers of servers are assumed to be 1 and 4 while assuming that the request generating nodes are four. In the simulation, it is considered that the locally served data is 40\% and 60\%.
It is evident from the figure, as the number of servers that can serve a particular request decreases throughput decreases and average delay increases. Considering the case of (S=1 and a=40%) and (S=1 and a=60%), it is evident that throughput decreases form 0.6 to 0.4 while average delay increases by one unit (figure 4.14). While considering that the available servers are four in number the throughput is one at all the loading conditions and average delay is nearly zero. This clearly indicates that to keep throughput at very higher level (~ 1) and average delay at zero level the replicated data should be available to comparatively large number of servers.
Case V: It is assumed that the numbers of servers that can serve the requests are more than one and the load is shared among the servers.

In this section, it has been analyzed that if load is distributed among the servers, then how delay will be reduced as throughput will surely be increased. Here, three cases have been considered for the load distribution as shown in figures (4.15 to 4.24). The simulation is run for $10^3$ slots and these figures also reflect the effect of the simulation on the overall results. In the earlier figures the simulation is run for $10^6$ slots. Here, there is large reduction in the number of slots i.e. $9\times10^5$.

In figure 4.15, the value of ‘a’ is considered to be 100% and 20%. It is clear from the figure that if the entire load is given to a particular server then at the higher load throughput decreases, and if load is shared and 20% of the load is on the server, the throughput always remains one.

![Figure 4.15. Throughput vs. load for B=4, N=S=4, and ‘a’=100%, ‘a’=20%.
](image)

Considering figure 4.16, it is also noticeable that when the entire load is given to a particular server then the average delay increases and attains the value of 4 slots at the load of 1. However, when the load is shared, the average delay at the load of 1 is of only
0.8 slots. It is noticeable that when numbers of slots are less, the delay increases linearly with load.

Figure 4.16. Delay vs. load for $B=4$, $N=S=4$, and ‘$a$’=20%, ‘$a$’=100%.

Figure 4.17 is very much similar to figure 4.15, here the value of ‘$a$’ is assumed to be 100% and 30%. It is clear from the figure that if the entire load is given to a particular server then at the higher load throughput decreases, and if load is shared and 30% of the load is on the server, the throughput always remains one. Hence, 30% of the load does not affect the throughput performance. Considering figure 4.18, it is also noticeable that when the entire load is given to a particular server than the average delay increases and attains the value of 4 slots at the load of 1. However when the load is shared, the average delay at the load of 1 is only 1.2 slots.

Figure 4.17. Throughput vs. load for $B=4$, $N=S=4$, and ‘$a$’=100%, ’$a$’=30%.

84
Figure 4.18. Delay vs. load for $B=4$, $N=S=4$, and ‘$a’=100\%$, ‘$a’=30\%$.

Figure 4.19. Throughput vs. load for $B=4$, $N=S=4$, and ‘$a’=100\%$, ‘$a’=50\%$.

Figure 4.20. Delay vs. load for $B=4$, $N=S=4$, and ‘$a’=100\%$, ‘$a’=50\%$. 
In figure, 4.19 the value of ‘a’ is considered to be 100% and 50%. It is clear from the figure, even for ‘a’ is 50%, the throughput performance remains the same as for a=20% and a=30%. Considering figure 4.20, the average delay at the load of 0.5 is one slot, and at the load of 1, average delay is of only 2 slots.

![Figure 4.21](image)

**Figure 4.21. Throughput vs. load for B=4, N=S=4, and ‘a’=100%, ’a’=25%.**

In figure, 4.21 the value of ‘a’ is considered to be 100% and 25%. It is clear from the figure, even for ‘a’ of 25%, the throughput performance remains the same as for a=20% and a=30%. Considering figure 4.22, the average delay at the load of 0.5 is 0.4 slot and at the load of 1, average delay is of only 1 slot.

In figure 4.23, the various values of value of ‘a’ are considered. It is clear from the figure if the entire load is given to a particular server then at the higher load throughput decreases, and if load sharing is increased the throughput starts improving. If the value of ‘a’ is 100% the throughput is 0.93 and when ‘a’ starts to decreases the throughput improves and finally attains a value of 100%. Considering figure 4.24, it is also noticeable that when the load increases, the average delay increases as the value of ‘a’ increases.
Figure 4.22. Average Delay vs. load for B=4, N=S=4, and ‘a’=100%, ’a’=25%.

Figure 4.23. Throughput vs. load for B=4, N=4, S=4, and with various values of ‘a’.

Figure 4.24. Delay vs. load for B=4, N=4, S=4, and with various values of ‘a’.
From figures 4.15 to 4.24, the main noticeable points are

1. If load is distributed then the throughput is very high and nearly one.
2. The distribution of load among various servers reduces the waiting time latency to a significantly low level.

Considering the figures from 4.15 to 4.20, where it has been considered that the load is shared between the three servers with the values of ‘a’ as 20%, 30% and 50%. The throughput is given by

\[ T = \frac{20}{100} T_{av}^S_1 + \frac{30}{100} T_{av}^S_2 + \frac{50}{100} T_{av}^S_3 = 0.2 \times 1 + 0.3 \times 1 + 0.5 \times 1 = 1.0 \]

And the average delay at load of 1 is given by

\[ D = 0.8 + 1.2 + 2.0 = 4.0 \text{ slots} \]

The above delay will occur only when one request is sent after another, but as it has been suggested once requests are generated they are shared among the servers. Hence, simultaneous request to various servers will reduce the delay and net delay is given by

\[ D = \max (0.8, 1.2, 2.0) = 2.0 \text{ slots}. \]

Considering figures 4.21 and 4.22, it is very clear that if the load is shared between four servers 25% to each, then the throughput will be 1 and average delay will be 1.0 slot. Similarly for the other values of ‘a’ the throughput and average delay can be obtained by examining figures 4.23 and 4.24.

Hence, it can be summarized that if the load is shared between the servers, then the throughput increases and average delay decreases. But this distribution of load will lead to the deployment of more number of servers or same data has to be placed on a large number of servers. The optimal value of the server that can maintain high
throughput and reasonable delay depends on various parameters like; type of networks, QoS, server capacity and request type etc. This analysis can be a part of further study.

Case VI: In previous cases it has been assumed that the generation of requests and service are random in nature, now if the generations of requests are relatively small and number of requests generating nodes are large in number then the probability function that at a particular server $K$ requests arrive is given below and is called as Poisson distribution.

$$P[K] = \frac{e^{-\lambda} \lambda^K}{K!}$$, where $\lambda = NP$,

In figure 4.25, the random traffic data is compared with the Poisson data. Here the number of request generating nodes and servers considered are four. It can be easily seen that in case of random traffic, as the load increases, the throughput decreases. However, in case of Poisson traffic, due to less number of arrivals, the throughput remains at one.

![Figure 4.25. Throughput vs. load for B=8, N=4, S=4, and considering both Random and Poisson traffics.](image)
Figure 4.26. Average Delay vs. load for $B=8$, $N=4$, $S=4$, and considering both Random and Poisson traffics.

In figure 4.26, the random traffic data is again compared with the Poisson data for average delay. Here the number of request generating nodes and servers considered are four. It can be easily seen that in case of random traffic, as the load increases, the delay increases. However, in case of Poisson traffic, due to the less number of arrivals, the average delay remains nearly zero at lower load and attains a value of 0.2 slot at load of 1.

Figure 4.27. Throughput vs. load for $B=4$, $B=8$, $N=4$, $S=4$, and considering Poisson traffic.

In figure 4.27, the throughput vs. load is plotted. Here, the number of request generating nodes and servers considered are four and the buffering capacity is varied
from 4 to 8 requests. It can be seen that the throughput is better in case of B=8 in comparison to B=4. However, in both the cases, due to the less number of arrivals the throughput is nearly one. In figure 4.28, the average delay vs. load is plotted. Here, the number of request generating nodes and servers considered are four and the buffering capacity is varied from 4 to 8 requests. It can be seen that the average delay is very similar in both the cases and it is very less ~0.27 slot.

![Average delay vs. load](image1)

**Figure 4.28.** Average delay vs. for B=4, B=8, N=4, S=4, and considering Poisson traffic.

![Throughput vs. load](image2)

**Figure 4.29.** Throughput vs. load for B=4 while varying N=S=2, N=S=4, N=S=8 and considering Poisson traffic.

In figure 4.29, the throughput vs. load is plotted. Here, the number of request generating nodes and servers vary from 2 to 8 and the buffering capacity is fixed at 4
requests. It can be seen from the figure, as the request generating nodes increases the throughput decreases. However, in both the cases, due to less number of arrivals the throughput is nearly one.

Figure 4.30. Average delay vs. load for $B=4$ while varying $N=S=2$, $N=S=4$, $N=S=8$ and considering Poisson traffic.

In figure 4.30, the average delay vs. load is plotted. Here, the number of request generating nodes and servers are varied and the buffering capacity is kept at 4 requests. It can be seen that the average delay increases as the request generating nodes increases and is 0.05 slot for $N=2$ and is of ~0.7 slot for $N=8$.

Figure 4.31. Throughput vs. load for $N=4$, $S=4$ while varying the values of $B$ and considering Poisson traffic.
In figure 4.31, the throughput vs. load is plotted. Here, the number of request generating nodes and servers are 4 and the buffering capacity is varied from 2 to 8 requests. It can be seen in the figure, as the buffering capacity increases the throughput increases.

![Figure 4.31. Throughput vs. Load](image_url)

**Figure 4.31. Throughput vs. Load.**

In figure 4.32, the average delay vs. load is plotted. Here, the number of request generating nodes and servers considered are four and the buffering capacity is varied from 2 to 8 requests. It can be seen that the average delay increases as the request generating nodes increase and is 0.54 slot for $B=2$ and ~0.78 slot for $B=8$.

![Figure 4.32. Average Delay vs. Load](image_url)

**Figure 4.32. Average delay vs. for N=4, S=4 while varying the values of B and considering Poisson traffic.**

In figure 4.32, the average delay vs. load is plotted. Here, the number of request generating nodes and servers considered are four and the buffering capacity is varied from 2 to 8 requests. It can be seen that the average delay increases as the request generating nodes increase and is 0.54 slot for $B=2$ and ~0.78 slot for $B=8$.

The results obtained under the Poisson traffic clearly show that, the throughput is nearly one with very less average delay. Hence, data distribution among the servers is not necessary. The Poisson traffic analysis is valid till date, the more data centric applications are coming up and in the near future the random traffic analysis needs to be considered for real time data replication processes.
In the above analysis, the server delay is taken into consideration. However, in WAN data may have to traverse to a geographically far node, in such a case the network delay becomes an important parameter. The network delay estimation is presented in the next section.

4.2 Estimation of Network Delay

When data requests traverse through the network, the network delay may be significant, and it becomes an important delay parameter. If the network delay \( D_{NW} \) is incorporated, then the total delay \( D \) can be evaluated as

\[
D = D_{av} + D_{av} + D_{NW}
\]

Now, with more data centric applications coming up, the whole computer network system is slowly transferring to fiber optic networks. Considering the fiber optic network, the latency in the network would be

\[
D_{NW} = \frac{L}{v} = \frac{Ln}{c} = \frac{100 \times 1000 \times 2}{3 \times 10^8}
\]

\[
D_{NW} = 0.67 \times 10^{-3} \approx 1 ms
\]

Where, \( L \) is distance between communicating end points, \( n \) is the refractive index of fiber and \( c \) is velocity of light.

If the global node is 100 Km away from the request generating node, then there will be a network delay of 1.0 ms and thus the round trip delay would be 2.0 ms. Similarly, if a local node is only 100 m away from the request generating node then the round trip network delay would be 2 \( \mu s \). It also introduces a latency that is proportional
to the size of the frame being transmitted and is inversely proportional to the bit rate as follows:

\[
D_f = \frac{F_s}{B_R}
\]

In the above equation, \(F_s\) is the frame size and \(B_R\) is the bit rate. For a frame of 64 bytes and data rate of 1000 Mbps, the delay is \(0.5\mu s\). As average queuing delay is in terms of frame size, for example a delay of 4 slots will be equal to \(0.5 \times 4 = 2.0\mu s\).

In case of global networks, the main contribution in the delay is due to the propagation delay. Considering the case, when all the generated requests are transferred to global networks (refer figure 4.4), the delay is

\[
D = D_{av}^G + D_{N/W} = 0.5 \times 5\mu s + 2.0ms \approx 2.0ms
\]

Again considering the case, when all the generated requests are served at local network (refer figure 4.4), the delay is

\[
D = D_{av}^L + D_{N/W} = 0.5 \times 5\mu s + 2\mu s = 4.5\mu s
\]

Thus, it can be observed that, on the overall delay the network delay plays a very significant role in the case of global networks. However, in case of a local network it plays a minor role in the overall delay.

4.3 Performance Comparision of the Proposed and the PDDRA Schemes

The PDDRA scheme mailly relies on the pre-fetching of the probable data to reduce the overall latency. In PDDRA scheme, required data is fetched form the master node for replication i.e., the internet. There is no provision to fetch data from the other nodes in the local network where data may be available. As in PDDRA scheme all the generated requests are sent to the global network, i.e. 100% of the load is given to the
global network. It fetches the required data along with the probable data that may be needed in future from global network in order to reduce the overall latency. As the bandwidth between RS and global network is lesser than that between VOs, it takes too much time to replicate the data as the existing PDDRA algorithm has to perform pre-fetching of probable data along with the required data.

However, in the proposed scheme a simultaneous request is sent to both local and global networks (master node) by taking into consideration that sometimes it may possible that data may be available locally at some other node.

In the following figures throughput and average delay of the proposed and PDDRA schemes have been given to analyze the comparative performance of both the schemes. Consider figure 4.33, here, it is considered that the number of request generating nodes are four and servers are also four and each server has a buffering capacity of 8 requests. This figure shows the simulation results of throughput with varying loads.

![Figure 4.33. Throughput vs. load for N=4, S=4, B=8 with ‘a’=20%, ‘a’=80%, ‘a’=100%.](image-url)
According to the PDDRA scheme, all the generated requests are sent to the global server; hence, ‘a’ which denotes the fraction of traffic being sent to the global server will be 100% i.e. ‘a’=100%. It can be seen that till load 0.7 both the schemes have same throughput, but as the load increases beyond the 0.7 mark, the throughput starts decreasing and attains the value of 0.96 at the load of 1.

In comparison to this, the proposed scheme also looks for the availability of the data at the local servers. If data is not available at local network then the proposed scheme has the same performance as the PDDRA scheme. However, if data is available at a local server, then the proposed scheme performs better than the PADRA scheme as shown in figure 4.33.

Now considering the value of ‘a’ as 20%, meaning 80% of the data can be fetched from the local servers. In such a case, the throughput of the proposed scheme is one irrespective of the load. Similarly, if the value of ‘a’ is considered to be 80%, meaning 20% of the data can be fetched from the local servers, and results are shown in the figure. Here, the throughput remains one till load 0.9 and it decreases slightly at the load of 1.0. In a nut shell both the schemes provide very high throughput even at the higher loads.

Another important performance criterion is the average delay. In figure 4.34, the average delay is plotted vs. load on the servers for both the schemes. Here, it is very clear that the average delay in PDDRA scheme is much larger in comparison to the proposed scheme. As the load crosses the 0.7 mark, the delay rises exponentially to attain a value of 5 slots at the load of 1.0.
However, with the proposed scheme while considering 80% of the traffic being transferred to the global servers the average delay is zero till 0.5 load and attains a value of one slot at the load of 1.0, which is very small in comparison to the PDDRA scheme. Similarly considering that 20% of the traffic being transferred to the global servers, in such a case the average delay remains zero for all the loads.

From figures 4.33 and 4.34, it is concluded that the throughput performance of both the proposed and the PDDRA schemes is nearly same, but on the higher loads the performance of the proposed scheme is slightly better than the PDDRA scheme. However, in case of average delay the proposed scheme performs much better than the PDDRA scheme as the PDDRA algorithm requires much time for mining the file access sequences and patterns for pre-fetching.