CHAPTER 6

VIPV – TRACKER SYSTEM

6.1 TRACKER SYSTEM

A proxy, called the tracker in our work, lies between the media server and the p2p clients. Its position in the VIPV system is presented in Chapter 3 and shown in Figure 3.2. It conveniently tracks the client requests that pass through it, as well as the network condition at the peer end. This intermediate and important position of the tracker, makes its operation lie on both sides of the VIPV system, i.e., the peer side and the server side. The coordination of the various schemes working in the system, and the proper maintenance of the tracker are crucial for the operation of the VIPV system. The co-operation provided by the peers, by uploading to the fellow peers with incentivize pattern as seen in Liang et al (2010), is modified and added in our work to motivate the peers in up-loading their buffered content. This induces all peers to be cheerful providers, as they are serviced with a greater download speed, for each fully completed or partially completed upload.

The peer manages the MRSM and the OPNS in such a way, that they are partly handled by the tracker. This is again another major role done by the tracker for the peers. The RSS is present in the tracker. Global cache management, and replacement of stored video content, are some important tasks performed by the tracker to satisfy the request made by the peers. The tracker computes the incentives, and also handles many hidden challenges in satisfying normal video requests and VCR requests.
The major services provided by the tracker to the media server, include projecting the bandwidth balanced placement of video contents, their generation of network coded data packets, content pushing and placing. These services help the tracker to balance the heterogeneity to a large extent. The basic processes of the tracker for all the combined effort is to be put into the router or a separate dedicated peer node. Here, it is implemented at the edge router of the system so that the tracker includes the following three major divisions in its modules split up; they are: i) the Stream Onlooker (SO) ii) the p2p Client helper (PCH) and iii) the Server-side Helper (SH).

The Stream Onlooker keeps a watch on the incoming requests from the client to the media server, and notes down the request information in the tracker’s Look Up Table (LUT). This LUT is provided as an initial entity in datasets for the prediction based learning algorithm. Also, it saves the video segment information stored in the media server during the push stream, which is used for managing further requests among the peer groups.

Besides all the other activities, it also watches the stream request generated within the peer group. As this module works jointly with the incentive calculation module and other modules alike, a peer node’s re-request for the same video content, is first allotted to a servicer. Once the supplier is assigned, figuring its incentive or penalty is crucial, and is performed by the tracker. If the servicer is subjected to penalization, the tracker also relevantly checks for congestion mishap, identified by the re-request sent by the same requestor, before properly settling the judgment. It also tries to reduce the congestion causing nodes amicably. The combination of these destined methods at the tracker provides novelty to this approach, with which we deem to have achieved far more than what was intended. These several modules working for the three divisions are projected in Figure 6.1, which makes VIPV a success.
The Server-side Helper (SH) is mainly concerned with scheduling the peer group requests, mostly at the tracker, and very few at the media server. The SH uses hashing to service the requesting peers. It maps the video segment with the requesting peers to identify the various peers stored across the peer group. The PCH and SO input the SH with the needed information for managing the admission control, dynamically altering the choice when the provider or the receiver is no more worthy of doing either servicing or receiving respectively, due to network congestion.

The topology of the peer group connected and maintained as ERMT with SFSS, gains more momentum in multicast streaming for the peers, as scheduled by the SH. The SH provides another prominent work to the server, that helps it to place the pushed video content at the peers, based on the ability of each peer. This is to identify the best supplier (peer) for the current VCR request provided through the AVL and splay trees. Several other modules that work for the benefit of the Server-side Helper are: Generation of Network Coded (NC) packets, Pushed Repository, Content Push, Replacement of stored video content, and Request Schedule Server (RSS).

**Figure 6.1 Detailed Design of Tracker System for working of the VIPV**
The Peer Client Helper (PCH) manages and provides information for Ant-based routing at the tracker. The Stream Onlooker (SO) updates and sends the information on each peer, based on the services required as well as the services provided. The PCH holds the latest information on the clips cached at each peer in all the peer nodes as a table. The various sub-modules working for the Peer Client Helper are: Request Schedule Server (RSS), Learning Algorithm, Peer Hit neighbor selector, Global Cache, Bandwidth aggregator and Incentive Calculator.

6.2 SERVER SUPPORT

The VIPV media server benefits more from the tracker side of the system with a majority of its complex works performed at the tracker and relieved of load. This so called relieving the media server is a requirement for the VIPV system, as the requests and streaming transit are minimized, and the server concentrates on new incoming requests and users, with the scaling of the system. The current socio network systems use this technique to ease the server from more huge and more complex tasks, such as managing heterogeneous clients with videos and video transmissions. As we concentrate more on the VCR solution, the dealings done by the tracker become even more relevant for the betterment of VCR operations.

The modules within the tracker, working for the benefit of the server, as pointed out earlier, are: Generation of Network Coded (NC) packets, Pushed Repository, Content Push, Replacement of stored video content, and RSS. Before we get into the working details of these various modules, the essence of the stream onlooker is provided in the next section 6.2.1.
6.2.1 Stream Onlooker

The Stream Onlooker (SO) is quite similar in a way to the analyzer module of the QPMVM. This keeps a watch on the incoming requests from the client to the media server, which notes down the request information in the tracker table (LUT). Also, it saves the video segment information stored by the media server during push stream, to be used for managing further requests among the peer groups. Among all the other things, it keeps watch on the stream request generated within the peer group, by looking for further requests to be made by the same requester, after assigning another peer to serve for the request currently made by the requesting peer.

For example, if peer ‘A’ has requested a video segment ‘i’, then the tracker could receive only two more requests for the same video segment from the same peer ‘A’. By this we guarantee that the request has either been serviced by another providing peer, for the better, or the request has not been served by the provider peer to whom the request was directed, for the worse, in which case, the tracker manages to route it to another peer and the media server simultaneously. In this way, the tracker fixes the servicing time for the requesting peer and looks out for another reliable provider with options open within the local peers of high profile range groups.

This part of the work at the tracker looks for capable service providers in the peer group that satisfy the requesting peer to get serviced, before the exhaustion of its buffered video content. Moreover, the agreed peer provider who would not serve the requesting peer client would get less benefit, in terms of speed and video quality as the agreement has been futile with no service at the client end. After the agreement, the non-providing peer node’s information could be captured with the same request, generated by the requesting client who has timed out its earlier request. This information on the non provider becomes fruitful, at the same time, to look for any further
requests from the non-provider, such as to look for network related problems like link congestion, attack driven malicious nodes etc., and if there are no such problems, the information would be used for negative incentives; otherwise, the supplier may be redeemed from negative incentives.

6.2.2 Network Coded Content

In this section we innovate and develop the working of a VoD streaming system, that uses NC for improving the delivered video content at the end-user, by correcting the error packets, and that achieves uninterrupted playback with efficiency in VCR operations.

The NC generator is present at the tracker for the peer clients, reducing the overhead at the server. The relevant packets that are lost within each peer-client, are generated with the NC packets. This is more helpful in correcting at a much faster pace, than the time consumed for retransmission. This also helps in improving user efficiency in VCR operations. Though the NC provides added advantage in P2VoD systems, there is initial transmission delay, a time cost incurred in video streaming. This time cost is rather small when compared to the difficulties within the Internet for the retransmissions.

In order to provide maximum video efficiency, throughput, error resilience and adaptability, the system is designed to withstand resilience, and be fault tolerant. And NC has been used to achieve the enhancement at the user end with ultimate benefits, by restraining retransmissions.

When the node is transmitting packets to other nodes, all the packets do not necessarily travel through the same channel. So each packet might end up travelling in a different channel. When they reach the destination, the quality of the video packets that have been received has to be ascertained. If any of the packets are lost in transmission, then the re-
transmission of these packets causes delay in the play back of the video or causes jittered video play. This is where the NC comes in handy.

We use the Linear Network Coding (LNC) technique over Random Network Coding (RNC), because it takes more time to encode and decode, and the complexity is more. The LNC helps to achieve more throughput in multicast networks. Since we try to provide an efficient VCR playback for the users, we have selected the LNC over RNC. The NC has gained momentum, due to the fact that it is robust, and theoretically claimed proof has provided ample opportunities.

Hence, our work on NC in media streaming with VCR functionality, that enhances performance through the tracker, is designed to generate network coded packets. This has not been dealt with so far in any of the similar P2VoD systems, such as UUSee, CoolStreaming, Joost etc. Moreover, our system working with the tracker, provides improved performance than the system proposed by Wang and Liu, 2008. The tracker identifies the packets to be NC and generates them. The NC has great potential, in terms of capacity, delay and resiliency to loss, in broadcast or multicast scenarios. It improves the system performance, while it avoids the use of complex routing or scheduling algorithms. It also necessitates a reduced control overhead in networks with diversity.

6.2.2.1 Generation of NC Packets

Without the NC, if some serving peers suddenly depart the system, some blocks of the original file or of the source-encoded file will disappear, and the remaining nodes will not be able to finish their downloads. This demonstrates that with the NC, nodes are able to finish their downloads even in extreme circumstances.
Here, we describe our model for end-system cooperative content distribution. This model can be used to either distribute blocks of the original file (no coding), or blocks of encoded information where the encoding can happen either only at the source (source coding), or both at the source and at the network (network coding). We will outline the basic operation of this system, emphasizing some algorithmic parameters.

In the current scenario, a network coder block has been placed in the tracker. The NC packets are transmitted to the requestor peer along with the original packets. If any of the packets are found missing from the original message at the receiver end, then we use the NC packets to reconstruct the lost packets. As we use the LNC here, the intermediate nodes are used to send messages from the source to the destination. The LNC, making use of the Hamming Code at source, is used here. In the Hamming Code, the processing of encoding the message is as follows. The video is converted to its binary format. The binary data is taken, and it is padded with some extra parity bits. A parity bit is placed at every $2^k$th location (i.e., 1, 2, 4, 8, etc., the value of $k$ ranges from 0 to the length of file-n). The value of $k$ is increased from 0 to n. This encoded data is transmitted.

At the receiver side, the parity is once again verified with every $2^k$ bit. If any bit is corrupted during transmission, then can be found and corrected. The data is then converted back to the video peer networks, whose working is explained in the following sections. The details of a peer such as: peer name, peer IP address, and the requested file name, are sent to the server communication unit through the tracker’s RTCP protocols. When the peer receives the video packets in response to a request, then the video packets are played in the media player, which has been designed to be stored in the local cache. The video files that get delivered to the peers are stored in the local cache. When the user wishes to play the video, then the video packets from this folder are taken and played in the media player.
The NC of the video packets at the different nodes can be explained as follows: Let, $P_i$ be the $i^{th}$ peer, Number of packets received by $P_i$ are $1...k...n$, viz., n packets. Let, $m$, be the number of parity bits in the $k^{th}$ packet, and the number of bits in packet are $2^m-m-1$. So, the numbers of bits that get transmitted are $2^m-1$.

When peer1 requests for a video available at peer2, the tracker transfers the request to peer2. Now, peer2 acts as the server and sends the video in packet form to peer1. Before sending from one peer to another, the video undergoes the LNC process. The NC packets are sent to the requestor peer along with the original message. During the transmission among peers, if there is any packet loss or some packets get corrupted, the packets can be regenerated by using the NC packets.

### 6.2.2.2 Evaluation Parameters

The evaluation parameters that have been considered for the evaluation of the system, are the response time, bit rate and packet loss recovery for normal video playback as well as for VCR playback. We implement our system with the available resources as mentioned in Table 6.1. The implementation done on these available peers has been studied, which would give more insight into the working of NC media streams.

The study with limited resources as shown in Table 6.1 does not make the system invalid, because here, we look for the changes that occur within limited peer groups, and are evaluated with comparison from non NC streams to NC streams. This highlights the part of our work, which brings about the need for NC not only for huge peer networks but also for the smaller peer networks. This provision is not dealt with by most of the researchers, trying to achieve benefits from NC. The different systems with which the evaluation was done are: (i) Normal p2p Network like BitTorrent,
(ii) p2p Network with NC (with Base64 video encoding) and (iii) p2p Network with NC (with Linear Coding).

### Table 6.1 Evaluation parameters

<table>
<thead>
<tr>
<th>Simulation parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Peers</td>
<td>10</td>
</tr>
<tr>
<td>Number of movies</td>
<td>5</td>
</tr>
<tr>
<td>Packet length</td>
<td>47-65 bps</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>256-2000 kbps</td>
</tr>
<tr>
<td>Simulation time</td>
<td>60 min</td>
</tr>
</tbody>
</table>

![Figure 6.2](image)

**Figure 6.2 Response time versus number of peers for normal play back**

For measuring the response time, the system run was done in all the different situations by increasing the number of peers. The response of the different systems is shown in Figure 6.2. In Table 6.2, during the testing for the module with NC, one packet was removed from the received video. The peer which received this data realized this missing packet, and decoded the NC packets for that missing packet alone. Hence, the response time of the module with NC function increases, as all the other processes, such as detecting the missing packet and then decoding it consumed more time. The
behavior of the system while operating upon different VCR operations at static intervals simultaneously by all peers, has been tabulated in Table 6.3.

When video requests are sent out at random time intervals which we mention here as dynamic VCR request time, the response of the system is on similar lines of the Static (regular time interval) response. In table 6.2, Peer5 in row-2, Peer8 in row-5, Peer9 in row-8 show the response times when the system performs NC. From these values, it becomes clear that the delay in transmission between the packets that have been regenerated through NC, and the ones that have been received unaltered, is 0.4% to 2%. This indicates that the NC streaming of video packets is better for even smaller peer network groups. It is highly possible, that if NC is incorporated by commercial structures to handle lower infrastructure groups they could benefit by this system. Further, Table 6.3 shows the various timestamps obtained for VCR operations with NC; this is more beneficial from the user’s perspective.

From the graph shown in Figure 6.2, we can infer that, the response time of the peer requests becomes more, as the number of requestor peers are increased. Also, when the amount of coding increases, in the following ascending order, p2p system, p2p with NC, we can notice that the response time also increases, as the processing time increases. Without NC, we can understand that the packet loss recovery extracted within the individual peer node system is nil. In such a case, it is mandatory for a Re-Request. The NC helps in effective playing of video despite some missing packets. From the values it becomes clear, that the delay in transmission between the packets that have been regenerated through NC, and the ones that have been received unaltered, is 0.4% to 2%.

Using a linear NC in Peer-to-Peer networks improves video efficiency, while performing VCR operations. The system is able to provide error-free video data to the end user quickly in spite of packet loss or corrupt
packets. We conclude that the NC can be implemented in P2VoD networks to enhance transmission. This is one of the major components in the VIPV system.

**Table 6.2 VCR Request Time (dynamic) and Playback time at different peers**

<table>
<thead>
<tr>
<th>VCR Request sent out time (ms)</th>
<th>Playback at Peer1</th>
<th>Peer2</th>
<th>Peer3</th>
<th>Peer4</th>
<th>Peer5</th>
<th>Peer6</th>
<th>Peer7</th>
<th>Peer8</th>
<th>Peer9</th>
<th>Peer10</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11.00</td>
<td>10.50</td>
<td>10.80</td>
<td>10.90</td>
<td>11.30</td>
<td>11.40</td>
<td>10.55</td>
<td>10.75</td>
<td>10.50</td>
<td>11.20</td>
</tr>
<tr>
<td>17</td>
<td>17.50</td>
<td>17.50</td>
<td>17.45</td>
<td>17.20</td>
<td>17.20</td>
<td>18.40</td>
<td>17.55</td>
<td>17.75</td>
<td>17.50</td>
<td>18.20</td>
</tr>
<tr>
<td>23</td>
<td>23.20</td>
<td>23.40</td>
<td>23.22</td>
<td>23.45</td>
<td>24.10</td>
<td>23.85</td>
<td>24.00</td>
<td>24.40</td>
<td>24.20</td>
<td>24.10</td>
</tr>
<tr>
<td>31</td>
<td>31.80</td>
<td>31.90</td>
<td>31.45</td>
<td>31.25</td>
<td>31.70</td>
<td>31.55</td>
<td>31.80</td>
<td>32.00</td>
<td>31.80</td>
<td>32.30</td>
</tr>
<tr>
<td>43</td>
<td>43.30</td>
<td>43.33</td>
<td>43.50</td>
<td>43.45</td>
<td>43.70</td>
<td>43.55</td>
<td>43.80</td>
<td>44.00</td>
<td>44.20</td>
<td>44.30</td>
</tr>
<tr>
<td>58</td>
<td>58.10</td>
<td>58.10</td>
<td>58.40</td>
<td>58.50</td>
<td>58.70</td>
<td>58.40</td>
<td>58.55</td>
<td>58.80</td>
<td>4.10</td>
<td>4.50</td>
</tr>
<tr>
<td>61</td>
<td>61.50</td>
<td>61.10</td>
<td>61.22</td>
<td>61.23</td>
<td>61.25</td>
<td>61.10</td>
<td>61.01</td>
<td>61.25</td>
<td>65.34</td>
<td>64.30</td>
</tr>
<tr>
<td>70</td>
<td>70.20</td>
<td>70.40</td>
<td>70.80</td>
<td>70.90</td>
<td>70.30</td>
<td>70.40</td>
<td>71.20</td>
<td>70.75</td>
<td>70.50</td>
<td>71.11</td>
</tr>
<tr>
<td>82</td>
<td>82.30</td>
<td>82.10</td>
<td>82.45</td>
<td>82.25</td>
<td>82.50</td>
<td>82.15</td>
<td>82.12</td>
<td>82.70</td>
<td>82.80</td>
<td>82.30</td>
</tr>
</tbody>
</table>

**Table 6.3 Response Times for various VCR operations**

<table>
<thead>
<tr>
<th>Fast Forward Request at</th>
<th>Response Time in seconds</th>
<th>Backward Seek Request at</th>
<th>Response Time in seconds</th>
<th>Random Seek Request at</th>
<th>Response Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 sec</td>
<td>2.71</td>
<td>5 sec</td>
<td>1.92</td>
<td>5 sec</td>
<td>3.5</td>
</tr>
<tr>
<td>1 min</td>
<td>3.21</td>
<td>1 min</td>
<td>2.87</td>
<td>1 min</td>
<td>4.67</td>
</tr>
<tr>
<td>5 min</td>
<td>6.17</td>
<td>5 min</td>
<td>5.01</td>
<td>5 min</td>
<td>5.25</td>
</tr>
<tr>
<td>10 min</td>
<td>9.5</td>
<td>10 min</td>
<td>6.35</td>
<td>10 min</td>
<td>8.1</td>
</tr>
<tr>
<td>15 min</td>
<td>11.51</td>
<td>15 min</td>
<td>7</td>
<td>15 min</td>
<td>4.5</td>
</tr>
</tbody>
</table>
6.2.3 Content Push and Placement

VoD users enjoy the flexibility of watching whatever video clips, whenever they want. The playbacks of the same video clip on different users are not synchronized. The tree-based p2p system was originally designed for this implementation. The tree-based overlay is synchronized, and receives the content in the order the server sends it out. This is fundamentally different from the requirement imposed by the VoD service. How to accommodate asynchronous users using the tree-based p2p system is a challenging design issue. The VCR functionality enables quick and user-friendly browsing of multimedia content, and offers users the interactivity that is lacking in the traditional video broadcasting service.

As seen in the related works, random seek requires extra bandwidth at the server and the underlying network. But, our VIPV system uses p2p functionality and provides VCR interactivity without overloading the server. And at the same time, it manages p2p in providing a fair placement of content, based on its strength and ability to upload and share. With our content push and placement mechanism, we maximize the delivered quality to individual peers, despite the heterogeneity and asymmetry of their access link bandwidth in VoD services. This also helps in supporting user interactivity with very minimum latency.

6.2.3.1 Content Push

The initial content placement increases the content availability and improves the use of the peer uplink bandwidth. Content placement and associated pull policies allow the optimal use of the uplink bandwidth. We eliminate the problems of peer node failure, by using the Network Code-based placement scheme. The most required video contents are proactively pushed to the peers during low network utilization. Each peer can offer a limited
number of services. If the number of requests that can be served is fewer than
the maximum number of content serve jobs at the target peer, the requests are
blocked or queued at the serving peers, and wait till they can start the
services. In content push, the total download rate must be greater than or
equal to the video play back rate. The assumptions made in this system are:

- Most required videos are proactively pushed to boxes during
  low network utilization
- Database tables act as set-top-boxes
- The parameters used are the server load, network load and client
  access time

The main modules involved in this project are listed as follows:

1. Content placement using full striping scheme
2. Pull the content in full striping scheme
3. Content placement using code-based placement scheme
4. Streaming video from the video server
5. Streaming video from the client

6.2.3.2 Content Placement using Network Code-Based Push Placement
Scheme

Each movie is chopped into windows of contiguous data. In the
code-based placement scheme, each window is divided into a smaller number
of data blocks, that are called source symbols. The source symbols are
encoded into a large number of blocks, by using the Rate-less Code algorithm,
and transferred using network coding. The large numbers of blocks thereby
generated, are called coded symbols. These coded symbols are pushed to the
entire peers during low network utilization time. This approach eliminates the
problem of peer failure.
6.2.3.3 Pull the Content in Code-Based Placement Scheme

In the pull phase, peers respond to user commands to play the content. A peer can generate a sub request which is sent to other peers in the list of suppliers. This is less than the total number of peers and later sends it to all the peers. All the peers forward the coded symbol of the requested video to the requesting peer. Then, windows are reconstructed by decoding the symbol at the target peers from all other peers.

6.2.3.4 Streaming Video from Server or Client

When the required video content is not available within the local peers, the peer clients establish connection with the server. In order to prefetch and store the content from the video server, ahead of the peer’s play out time, it uses the distributed prefetching protocol. If one client requests the same video, that client is connected to another client, while another third client would stream the video from the video server. If any content is missing during streaming, then simultaneously the NC packets regenerate the missing packets and relieve the connection with the server. If client1 departs from the network, then client2 discovers a new client for downloading a video within the limited time; otherwise it is connected to the server. Here, we alternately mention peers and clients who perform the client side tasks.

In the Push-to-Peer VoD system, the video is first pushed to a population of peers. This first step is performed under the provider or content owner control, and can be performed during times of low network utilization. Following the push phase, peers seeking a specific content pull the content of interest from other peers, as in a traditional p2p system. The Push-to-Peer approach is well-suited to the cooperative distribution of the stored video among peer networks where there is provider control. The Push-to-Peer approach is more generally applicable to cases, in which the peers are long-
lived and willing to have content proactively pushed to them, before the video
distribution among the cooperating peers begins. We consider a controlled
environment, with a set of always switched on peers, constant available
bandwidth among the peers, and the possibility of a centralized control using
tracker.

6.2.3.5 Bandwidth Oriented Content Placement – Fair Placement
Policy

Our system utilizes the large storage capacity of peers to amplify
the supply of videos, so as to easily support the large demand in a scalable
manner. In our proposed scheme, the p2p enjoys our innovative Fair
Placement Policy (FPP), which is used to place a large number of video
blocks around the p2p network. Here, videos are divided into smaller
segments, and stored in peers distributed, based on the individual bandwidths.
A video mesh is built upon peers to support playback, and jumping FF/BW
during playback. A downloader on initialization contacts the tracker, sending
information like name of the file it is downloading. The tracker then responds
with a list of active peers, which has also downloaded the same file. Based on
this, the downloader then connects to the respective peers in order to receive
the required content. The role of the tracker is essentially limited to assisting
peers to find one another and keeping statistics, and thus the load on it as
minimal.

Once the media server forwarded the request to the tracker, then the
tracker acquires the nearest content replicated peer information, and sends the
request for the content. The server part of the peer maintains the cache and
stores the contents which have a high hit rate, and deletes the content which
has a least or no hit rate. The reduction in the download time of each client
using peer sharing, is handled here by means of the reduction in traffic load
on Internet links and ISPs, and the reduction in the wastage of resources like
bandwidth due to redundant packets. In addition, the system achieves a very low start-up and seeking latency, which are crucial for the performance of an interactive VoD system.

Also, the p2p application server receives the upload/download link capacity and the stream availability information from each peer. With the above information, the p2p application server suggests a peering list for the newly joining peer. The peer linking considers the routing cost, the congestion level of the links, the end-to-end delay, the matching of the upload/download link capacity, and the provision of the stable streaming service.

In order to offer quality streaming services, streaming servers are required to process multimedia data under timing constraints. In the traditional client-server architecture, when the number of clients increases, too much of server bandwidth, and the bandwidth of the access routers is wasted. This leads to each client getting low download rates and bad user experience. We propose a Fair placement policy combined with an incentive calculator, for the p2p system in multimedia streaming services.

Here, peers collaborate to form a distributed system, for the purpose of exchanging the content. It induces the cooperation between p2p systems and servers, to achieve an efficient and fair usage of network resources. A file that one peer downloads is often made available for upload at other peers. In the proposed scheme, the p2p provides a place for a large number of video servers around the Internet. This enables the end users to obtain a streaming video from one of the nearby servers, thus reducing the end-to-end delay and overall network congestion. With,

\[ m \] - total number of peers in the system
\[ N \] - total number of segments a video is split into
\[ A \] - average bandwidth of all \( m \) peers
b_i  - bandwidth of ith peer ‘P_i’
ph  - high bandwidth peers
pl  - low bandwidth peers
N_pl - Total number of segments to be allocated for low bandwidth peers
N_ph - Total number of segments to be allocated for high bandwidth peers

1. Categorize the m peers into ph and pl
   1a. ph where, bandwidth > A for all nodes in this set and
   1b. pl where, bandwidth < A for all nodes in this set

2. While still ‘N’ to be placed among m peers ‘P_m’
   a. N_ph = (N * (b_i) PeerHigh) / (A*m)
   b. N_pl = N - N_ph
   c. N_i = (N * b_i) / (A*m)
   d. N_i video segments are distributed to P_i.

This placement algorithm tries to place a larger number of video segments at higher bandwidth client side, and lesser number of video segments at lower bandwidth client side. According to the identification of the peer clients, ph serves as higher bandwidth and pl with a lower bandwidth in a heterogeneous environment. For each movie, the available clients with the two categories are identified, and the placement of the movie segments is performed by choosing the proper peer.

The video segments that have been obtained by splitting the video file are distributed among the various peers in the system, based upon their utility bandwidth. More precisely, the peers having more bandwidth receive more segments and vice versa. It is also possible for the end user to view a specified segment in the video. This improves user interactivity of the system by providing a random-seek option.
Ensuring fairness among the peers is done implementing limited Contribution Policies; i.e., each peer will fulfill its contribution contract, in terms of both its committed streaming rate, and the number of streaming sessions to serve. Managing high connection setup and media delivery latency, i.e., the efficient selection of supplying peers, helps in achieving the minimum buffering delay in the consequent peer-to-peer streaming session. This work is novel, as we contribute a faster system with its storage capacity, i.e., the clients that received the video content must contribute the video to the needing peers.

6.2.4 Replacement of Stored video content

The replacement of client held video clips is another function performed by the tracker. The solution for clip replacement is simple, and represents work at complex times such as handling of transient peers also. Apart from that, the content replacement performs the proper management of new contents to be pushed at the clients, by connecting to the content push manager within the tracker module.

6.2.4.1 Cache Content Replacement

The working of the content replacement in the cache is held with a Timestamp ‘T’, which is maintained in the replacement module in the tracker. The Globally checked LRU Algorithm is used. All the clients are checked for the least recently used movie. The segments of the LRU movie stored are later replaced. The Timestamp array holds the latest timestamp of the films that are requested at the tracker. A timestamp per video is maintained, that helps in replacement.
6.2.4.2 Client Buffer Management (Cache Maintenance)

During patching and caching, a client will receive data at a rate of 2R bps while playing back video at R bps. Thus, the video data will accumulate in the client buffer at a rate of R bps. To determine the maximum client buffer requirement, we note that the video data accumulates in the client buffer at R bps, when a client is performing patching and caching. During that phase, t seconds of video is received in t/2 seconds. L is the length of the movie (sec). Since $t \leq L$ the maximum buffer size required is equal to half of the video, i.e. $RL/2$ bits.

With the client buffer, we can further optimize the PAUSE operation. Specifically, instead of immediately breaking away from the video stream, once a PAUSE operation is executed, the client can continue to receive and cache the video data at a rate of 2R bps, if patching and caching is in progress or R bps. Thus, with a buffer size of $B_c$ bits, no matter whether a client is in the progress of patching and caching or not, it can implement a resource-free PAUSE operation if, $T_p + T_{pc} \leq B_c$ where, $T_p$ and $T_{pc}$ are the durations of ‘pause’ and ‘patching’ and caching, respectively. Otherwise, the client will issue a merging request to merge back to an existing full stream.

6.2.5 Request Schedule Server

The RSS is one main module, next to the Stream Onlooker in the tracker, that handles streaming requests from clients, especially requests arising within peer groups. It indicates the best place to receive the clips from. The new-movie request and the interactive playback are handled for scheduling at the RSS, which is present at the tracker. The RSS here manages almost all the requests subjected to the peer group, that is authenticated by the
media server. As for any tracker systems, such as the Bittorrent or UUSee, the main theme lies behind the installation of the RSS for streaming requests.

Other factors of the media server are, it enrolls fresh peers to the client community; details about the new client are flashed to the RSS, from where the RSS takes over. The media server also modifies the multicast streaming pattern appropriate to the existing peers, which it receives from the RSS through round the clock feedback. The RSS does the task of notifying the residence of the requested video clips, among the peer client groups. A VCR request made by any of them is found to be very easy and efficient, when serviced by the RSS. The network congestion and the server are relieved by the process, with the information to the tracker, which responds to the request for VCR operations, only when the corresponding video clips are not present in any of the client cache. Applying the VCR functionality in this environment is effective.

The clients in the distributed p2p client group are connected among themselves, and each client is further connected to the RSS separately. All the clients can interact with the media server through the tracker’s RSS. Scheduling is an important aspect here, where it makes admission control for the multicast peer groups. As we have seen, there are three different types of requests which fall in to one of the categories; New_Video Request, Same_Video Request and VCR Request. To select a queue for servicing, we have the First Come First Served (FCFS) and Maximum Queue Length (MQL), the two commonly used policies. The FCFS selects the queue with the oldest request, whereas MQL selects the longest queue. The MQL yields a higher throughput and is used here.

As all the different requests fall into any of these three types, three different queues are maintained appropriately. The New_Video Request, is handled using the Weighted Queue Length (WQL), where the weight for the
client is assigned with the waiting time at the queue. As two or more clients ask for the same video they are put in the same queue if already present in the New_Video Request. The more clients that request a new movie, the faster it would be served. This is the WQL servicing scheduling pattern.

For the Same_Video Request, as we can see, there is already a Queue under New_Video Request with a different movie title. The RSS merges the requesting client with patches of the missing streaming video clips, and sends the full stream from the already started stream, with information from the SFSS. The merging of the stream is provided with the details that the ERMT provides.

The request for a clip is processed by maintaining a look up table. The LUT stores the information about the clients and the video clips. The information about the client includes its IP Address, port number, client_ID, cache size_permament, cache size_temporary etc. The details about the movie clips present in the permanent cache are stored in the LUT. As per the request for a clip, the RSS checks the LUT for any entry of the video clip in any client, and acts accordingly by transferring the request to the client holding the clip. If not, the media server is requested. The LUT is implemented as a hash table, with the port number of the clients as the key to the elements stored in the table.

6.3 PEER SUPPORT

In the P2VoD peers share a number of resources, and the problem with this system is that, the peers rarely connect with each other for longer hours. Finally, there is a general problem of fairness in load distribution (Adar et al. 2000). The total download rate across all downloader must, of mathematical necessity, be equal to the total upload rate.
The strategy for allocating the upload, which seems the most likely to make peers happy with their download rates, is to make each peer’s download rate proportional to their upload rate. In practice it is very difficult to keep the peer download rates at a constant level. Sometimes dropping to zero by chance and much less, requires the upload and download rates be correlated. It is solved by the dynamic adaptation of incentives, followed for uploading to fellow peers for downloads received from those peers. We shall discuss this incentive mechanism in the following sub sections.

6.3.1 Fine Tuned ‘tit-for-tat’ Incentive Management

As presented in section 5.1.3, the peer incentive is combined with the peer side module Peer Manager and the tracker’s Incentive Calculator. This provides the proper incentive, calculated using the amount of perfectly uploaded service done by a certain peer, in order to receive the incentive. The coordination between the PCH and SO is required for managing the incentives and stream with the bandwidth aggregator, that combines multiple streams in order to provide high speed data to the servicing clients.

The p2p Client Helper (PCH) manages and provides information for Ant-based routing at the tracker. The Stream Onlooker (SO) updates and sends the information on each peer, based on the services required as well as services provided among peers to the PCH. The PCH holds the latest information on the clips cached at each peer in all the peer nodes, as a table. The Tracker stores the up-loader’s peer information for incentive management. Combining Multiple Up-loaders is another component initiated here at the tracker, called Bandwidth Aggregator.

**Bandwidth Aggregator:** This is done as one of the group members say ‘Pi’ who has maximal bandwidth, say ‘q’ kbps, but currently provides at the rate of ‘q-x’ kbps (since in reality, complete bandwidth is underutilized); this ‘x’
kbps would be compensated by another peer in the group by sending at its current bit rate ‘plus’ the left out bit rate; that is, if it sends at ‘r’ kbps and its maximal is below ‘r + x’ kbps, it compensates the complete streaming rate. If not, the ‘x’ kbps would be shared among the existing group, to provide incremental support bandwidth, which is obtained by adding the current sending rate plus 2 kbps by all, until the receiver is fulfilled with the added bit rate. The PM collaborates with the tracker in maintaining peers with proper incentives. This achieves high delivery to the client side, by combining these several solutions.

6.3.2 Peer Transient Management

The loss of quality in video streaming in p2p networks, due to the frequent loss of intermittent nodes (transient nodes) due to the failure to serve promptly, and node departure has been the primary motivation behind this study. Despite the increasing buffering time, this loss had been significantly affecting the quality of the video. Hence, this prompted us to include in the system the function that not only handles transient nodes, but also dynamically adjusts the network structure in the event of a failure. This ensures optimum efficiency, and at the same time, enhanced reliability.

Hence, we propose a new Transient Resilient Protocol (TRP) to assure an alternative path, to accelerate the data dissemination process under network churn due to transient nodes. Our work on VIPV focuses to assure an alternative path in the case of node failure, and data proximity during the video file request. The video file is stored and then played (store and play method). These versions would be transcoded and sent to the clients, without compromising much on the picture quality, and using storage memory efficiently. In this module, we mainly focus on promoting proximity awareness, fast data dissemination, and ensure resilience to node failures.
The P2VoD network is the best means to ensure enhanced performance. It also helps in fast streaming of videos. The TRP aims to assure an alternate path, and data proximity to accelerate the data dissemination process under network node transients. The TRP enabled overlay design promotes proximity awareness, dynamic load balancing, and resilience to network churns by node failures. The existing systems handle neither the proximity awareness condition nor the node failure scenario. However, such approaches sacrifice the quality of the video, besides being costly affairs. The proposed architecture overcomes the problems in traditional solutions, by incorporating node failure handling and by checking proximity.

We consider dynamic p2p networks under extreme transient conditions. Our goal is to achieve maximum throughput and fast data dissemination rate from any peer, with a small overhead even under frequent node failures. The heartbeat message will change accordingly, when a node departs or joins the network. The resilience and recovery process helps in altering the network structure under these conditions, performing the dynamic load balancing, as was not done in Tapse et al (2011).

Checking the proximity helps in finding the optimized peer for the transfer of a video file once the request is sent. It enables to find the node at maximum proximity (both with respect to network closeness and bandwidth capacity). This ensures the fast dissemination of data. The network proximity is measured by the latency. This proximity enables faster transferring of data items. The capacity proximity is measured by the closeness of the nodes with respect to node capacities. The capacity is measured by the number of nodes to which the node is connected, as mentioned in Li et al (2011). It also makes sure that the cache is large enough to hold the video file. The proximity includes network proximity, capacity proximity, and checking the cache parameters. The following conditions are imposed in TRP.
a. Each peer should send the heartbeat messages to its neighbor.

b. The peer that is departing should not send a heartbeat message, which is an indication that the peer is leaving the network.

c. Connection should be established once again if the peer moves in.

Every peer sends heartbeat messages within every time slot, when involved in receiving or sending streams. Once a peer makes a client request for a video file, the transient resilient process is activated. The check proximity module is executed. This check for the maximum proximity peer in terms of network, capacity, and cache size, provides its compatibility for the storage of the video file segments. Then, the maximum proximity determined peer will send the video file to the client peer.

If a peer leaves the network, i.e., the absence of the heartbeat message is observed, the Resilience and Recovery process comes into the picture which helps in maintaining the network structure and provides dynamic load balancing. It makes sure, that if any node has a correct successor it can be recovered from node failure. It then checks for the peer having the video file with maximum proximity. Then the new video server peer is found, which will transmit the requested video file. When no node departs, only the check proximity module is activated in the TRP process.

When a node failure occurs, there is an absence of heartbeat messages. The resilience process alters the network structure. The check proximity looks for the maximum proximity peer with the video file, thus ensuring fast data dissemination.

6.3.2.1 TRP process

This is the major module in the tracker. It continuously updates the network, according to the chord link weight, proximity weight and adaptation
interval. It performs two operations – connection and adjustment. Connection builds more chord links. The node periodically calls for the adjustment operation to replace the high-weight links with low-weight ones. If \( d(x,y) \) is the link’s one-way latency, \( c(x,y) \) is the capacity difference between nodes \( x \) and \( y \), and \( \text{suc}_x \) is \( x \)’s successor, then,

\[
\text{Network proximity weight, NPW} = \frac{d(x, y)}{d(x, \text{suc}_x)} \tag{6.1}
\]

\[
\text{Capacity proximity weight, CPW} = \frac{c(x, y)}{c(x, \text{suc}_x)} \tag{6.2}
\]

\[
\text{Proximity weight} = \text{CPW} \times \alpha + \text{NPW} \times (1-\alpha) \tag{6.3}
\]

where, \( 0 < \alpha < 1 \)

The Transient Resilient Process is the main process which helps in the simulation of handling node failure, and to ensure proximity awareness and dynamic load balancing. Proximity awareness ensures that the requested video file is downloaded from the maximum proximity server. The proximity is measured in terms of network, and capacity, along with cache parameters, so that the alteration happens on the fitting nodes alone.

a) Network Proximity: The neighboring nodes are identified by the heartbeat messages. This ensures fast dissemination of data. The video server sending a message with minimum latency will be chosen for the transmission of data. Initially, if a request has been received, it will check in the database for the suitability of the data. If there is no file available, then the request will be forwarded to its neighbors. The neighbors will check if the file is available in their own databases. Then the forwarding / transferring process goes on.

b) Capacity Proximity: This is measured by the closeness in node capacities. The node capacity is the maximum number of nodes to which it can forward the data item concurrently; i.e., the number of neighboring nodes or number of nodes to which the particular node is connected. In our structure, a node
has the maximum of four edges leading to its neighbors. If the peer with the video file is found by the network proximity, then the check proximity module is executed which checks for the capacity of the peer with the video file, and the one that requested the video file. If the capacities are approximately equal, then the peer is allowed to transmit the video file.

If \( x \) and \( y \) are based on the bandwidth capacities of any two peers \( p_1 \) and \( p_2 \), with their lower capacity encountered so far, for the allotted higher capacity, then,

\[
\text{Capacity closeness} = \frac{\text{smaller value between } x 	ext{ and } y}{\text{bigger value}} = \frac{x}{y}
\]

(6.4)

\( x \) and \( y \) are the data on proximity from random peers, with one of the requesting peers \( p_1 \) with value ‘\( x \)’, and the other peer \( p_2 \), with video file ‘\( y \)’. Also, \( x \) having smaller capacity at the requesting peer and \( y \) having larger capacity at the serving peer,

\[
\text{capacity proximity} = \begin{cases} 
  x:y = \text{close}, & z \geq 0.5 \\
  x:y = \text{not-close}, & \text{otherwise}
\end{cases}
\]

(6.5)

where, \( z = \text{Closeness in capacities.} \)

If the closeness value is greater than or equal to 0.5, then the peers are said to have closeness in capacities, and it is sent to check the cache size.

c) Cache Parameters: The peer is checked for the cache size, and if the cache size is good enough to hold the segmented video file, then the peer is chosen. The free cache size of the requested peer is sent along with the name of the video file. It is retrieved from the request message forwarded by the peers. The free cache size is checked with the size of the video file. If the cache size is bigger, then the video file is transmitted. For the download
request by the requesting peer, the node is checked for the availability of the free cache size to hold the video file.

Here, it checks for the cache size, and if the cache size is good enough to hold the split file, then the peer is chosen. Along with the network and capacity proximity, if free cache size proximity of the video file is identified, transmit the video file. Otherwise, forward the request to its neighbours. The cache size for every peer is assumed here. The cache size ranges from 2-3 MB. Since the executing environment is the same for more than one peer, the cache size is assumed for all the peers.

6.3.2.2 Resilience on Transient Nodes

Once a node receives a departure notification from its parent or detects the failure of its parent by heartbeat messages, it actively switches its tree parent. The recovery is done when a node on the ring of a TRP-enabled overlay keeps a correct successor, the data can be recovered from any node failure.

Input: node y, y’s neighborhood, the connection message initialized by node x, x’s degree
Output: New chord links to be established between node x and other existing nodes
Process:
where,

\[ R(x) = \text{The set of } x\text{’s neighbors} \]
\[ \text{PW}(x,y)=\text{Proximity weight between } x \text{ and } y \text{ nodes.} \]
\[ D_y=\text{Threshold degree of } y \text{ to build links.} \]
\[ d_y=\text{Node degree of node } y, \ d_y<=c_y. \]
\[ c_y=\text{Capacity of node } y. \]
\[ g_y = \text{cache parameter showing availability,} \]
\[ \text{Min } = \text{minimum cache freeness to fit video} \]
\[ y_f = \text{Node } y\text{'s farthest neighbor with highest chord link weight} \]

1. if \( x \in R(y) \) then goto Step 7
2. if \( d_y < D_y \) then Establish a chord link between \( x \) and \( y \)
3. else, if \( PW(y, x) < PW(y, y_f) \) then
   4. if \( D_y \leq d_y < c_y \&\& \ g_y \geq \text{Min} \) then establish a chord link between \( x \) and \( y \)
5. else if \( x \) not belongs to \( R(y_f) \) and \( d_x + 2 \leq c_x \) then
   Remove the chord link between \( y \) and \( y_f \)
   Establish a chord link between \( x \) and \( y \)
6. Establish a chord link between \( x \) and \( y_f \)
7. Else forward the message randomly to a neighbor of \( y \)

6.3.2.3 Performance of TRP

Checking all types of proximity helps in finding the optimized peer for the transfer of the video file. It enables to find the node at the maximum proximity (with respect to network, capacity and cache parameters). This ensures fast dissemination of data. The network proximity is measured by the latency. This proximity enables faster transferring of data items. The capacity proximity looks for the nodes having closeness of node capacities. The cache ensures that data could be held for further transfers.

From Figure 6.3, it is known that if there is a heartbeat message from where the neighbors’ information is available, the structure can be altered by the TRP process, which alters the heartbeat message sent, and thus makes a change in the network structure. Therefore, it is evident that only minimal time is taken for the resilience process to alter the structure, in the case of any node failures or network anomalies. The time taken to alter the structure if \( p_4 \) and \( p_{16} \) depart, along with the sending time for the video file,
is shown in Figure 6.3. This is shown in red colour for peer4 and green color for peer16. With peer4 departure, the delay is not much as the video content is held in the local cache of the receiving peer, and is supported by peer5 instantaneously. This graphical representation in Figure 6.3 highlights the check proximity failure with peer7 and peer9 not supporting the video file transfer. Later, peer10 helps in the resilience function when peer16 departs, although with lower capacity, and hence, with delayed time.

Figure 6.3 Illustration of ‘Resilience after Check Proximity’ which alters the network structure and sends the requested video file

Figure 6.4 Time taken to alter the peer structure from the departing time

Check proximity looks for the node of maximum proximity with respect to network, capacity and cache. The node should have a cache size
that can hold the video file; this is the entity that was not present in the CRP model (Li et al 2011). Figure 6.4, illustrates the departing peers, and their relevant TRP processing time, with each node in the peer leaving the system. Our work stands apart from other P2VoD systems that omit the transient behavior in peers. We make it a point to handle the resilience, and provide a far better P2VoD, the VIPV in our work.

6.3.3 Neighbor Search – Learning Algorithm

The overlay network of p2p systems should be dynamic and on-demand. Also, these on-demand connections should be such, that they increase the throughput of the system. Random connections will not be of any assistance. A predetermined logic should be used to help make the dynamic connections to increase the overall system efficiency.

We build an adaptive system that learns to predict neighbors, from the past known transactions of the requestors and servicers, so that the peer neighbor’s selections are optimized. The past transactions are held here at the tracker, which provides the learning algorithm that is used in AGILE based neighbor selection. This is implemented, as mentioned in the previous chapter, section 5.3.3.1, to benefit the VoD user to the maximum limit. Our work is supported by several blocks in the tracker module, that does learning followed by network coding.

The learning algorithm is linked with the pushed content repository and the global cache, that deals with the archive of information to support the learning algorithm as well. With the stored information, the provision of more accurate information in the neighbor information of a particular peer client node is provided by the AGILE system, as explained in the previous chapter’s section 5.3.3. Moreover, changes in the system are routed back through the feedback system that tries out new ways of understanding, to generate
requests only to competent neighbors and not to all neighbors. This is done by the peer hit neighbor selector here at the tracker, that provides information along with the streamed content.

As the information on learning algorithms is provided in section 5.3.3.1 (d) in the previous chapter, here we have given other modules that combine for the better working of the peer systems, that is helped by the tracker to a larger extent than what is perceived.

6.4 SUMMARY

Tracker based systems have an enormous opportunity to enhance the VIPV process, and remove the unintended factors. Some of the solutions provided with the help of tracker are:

- Avoids retransmission of lost packets by using NC. Thereby, the response time is less, when compared to systems that make retransmissions for any missing packets. As retransmission is not necessary, the network congestion is eased in larger peer networks and offers lesser overhead.

- The TRP included in the VIPV tracker’s ultimate goal is to achieve sustained throughput and fast data dissemination rate from any data source with a small overhead even under frequent node failures. This system helps in fast and reliable dissemination of data, which is achieved through proximity (network, capacity and cache). This also recovers a network from frequent node failures through resilience.

- Next, the tracker provides ample amount of help in reducing the congestion rate among the peer links, thereby adding to the efficiency of the video streaming process.
• The incentives providing scheme for peer nodes gives higher download bandwidth to the nodes, which upload and share more to the peers, that helps further in increasing the video streaming rate.

• We observe that content push and placement along with cache replacement greatly enhances the activities shared by the tracker in reducing the server side load, that provides users with good quality video. All peer nodes should be synchronized and should contain almost the same content, so that the user side reception is good. Achieving this using the tracker is mastered in a video streaming environment.

The results have proven that the tracker is beneficial in general for VoD operations, and especially for VCR operations, such as fast-forward, pause, replay, backward, random seek etc., in the VIPV.