A Framework For Efficient Data Distribution In Peer-to-peer Networks.

Darshan Purandare
University of Central Florida

A FRAMEWORK FOR EFFICIENT DATA DISTRIBUTION IN PEER-TO-PEER NETWORKS

by

DARSHAN PURANDARE
M.S. University of Central Florida, 2005
B.S National Institute of Technology, Bhopal, India, 2001

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2008

Major Professor: Ratan Kumar Guha

ABSTRACT

Peer to Peer (P2P) models are based on user altruism, wherein a user shares its content with other users in the pool and it also has an interest in the content of the other nodes. Most P2P systems in their current form are not fair in terms of the content served by a peer and the service obtained from swarm. Most systems suffer from free rider's problem where many high uplink capacity peers contribute much more than they should while many others get a free ride for downloading the content. This leaves high capacity nodes with very little or no motivation to contribute. Many times such resourceful nodes exit the swarm or don't even participate. The whole scenario is unfavorable and disappointing for P2P networks in general, where participation is a must and a very important feature. As the number of users increases in the swarm, the swarm becomes robust and scalable. Other important issues in the present day P2P system are below optimal Quality of Service (QoS) in terms of download time, end-to-end latency and jitter rate, uplink utilization, excessive cross ISP traffic, security and cheating threats etc. These current day problems in P2P networks serve as a motivation for present work. To this end, we present an efficient data distribution framework in Peer-to-Peer (P2P) networks for media streaming and file sharing domain.

The experiments with our model, an alliance based peering scheme for media streaming, show that such a scheme distributes data to the swarm members in a near-optimal way. Alliances are small groups of nodes that share data and other vital information for symbiotic association. We show that alliance formation is a loosely coupled and an effective way to organize the peers and our model maps to a small world network, which form efficient overlay structures and are robust to network perturbations such as churn. We present a comparative simulation based study of our model with CoolStreaming/DONet (a popular model) and present a quantitative performance evaluation. Simulation results show that our model scales well under varying workloads and conditions, delivers near optimal levels of QoS, reduces cross ISP traffic considerably and for most cases, performs at par or even better than Cool-Streaming/DONet.

In the next phase of our work, we focussed on BitTorrent P2P model as it the most widely used file sharing protocol. Many studies in academia and industry have shown that though BitTorrent scales very well but is far from optimal in terms of fairness to end users, download time and uplink utilization. Furthermore, random peering and data distribution in such model lead to sub-optimal performance. Lately, new breed of BitTorrent clients like BitTyrant have shown successful strategic attacks against BitTorrent. Strategic peers configure the BitTorrent client software such that for very less or no contribution, they can obtain good download speeds. Such strategic nodes exploit the altruism in the swarm and consume resources at the expense of other honest nodes and create an unfair swarm. More unfairness is generated in the swarm with the presence of heterogeneous bandwidth nodes. We investigate and propose a new token-based anti-strategic policy that could be used in BitTorrent to minimize the free-riding by strategic clients. We also proposed other policies against strategic attacks that include using a smart tracker that denies the request of strategic clients for peer list multiple times, and black listing the non-behaving nodes that do not follow the protocol policies. These policies help to stop the strategic behavior of peers to a large extent and improve overall system performance. We also quantify and validate the benefits of using bandwidth peer matching policy. Our simulations results show that with the above proposed changes, uplink utilization and mean download time in BitTorrent network improves considerably. It leaves strategic clients with little or no incentive to behave greedily. This reduces free riding and creates fairer swarm with very little computational overhead. Finally, we show that our model is self healing model where user behavior changes from selfish to altruistic in the presence of the aforementioned policies.

Dedicated to my dear parents, Mrs. Manjusha and Dr. Shrinath Purandare.

ACKNOWLEDGMENTS

I wish to acknowledge the support of all my thesis committee members for their help throughout the PhD program. In particular, Dr. Guha who has been a mentor, advisor and also a great friend. He always gave me the freedom to carry out research on the subject I liked. His guidance and support, both academically and personally, has been pivotal to my career. I am very grateful to him for providing me the opportunity to pursue this work. He supported me throughout the Ph.D program and also sponsored overseas travels for various conferences.

I wish to thank my parents, Dr. Shrinath Purandare and Mrs. Manjusha Purandare for their love and support all throughout. It would not have been possible without them. My grand father, Kamlakar Purandare, always encouraged me to pursue Ph.D and today I want to thank him for his vision, and moral support. My wife Gautami has been very loving, supportive and encouraging during the whole course of my PhD. She knows quite a lot about my research as she helped me at every stage that includes brainstorming, proof reading, creating figures for papers and the list goes on. My sister Amruta , brother-in-law Ronit, her two little kids, Rutvik and Aaryan deserve a special mention for their continued love and support. Special thanks to Dr. Baba, Hema Shirhatti, and Aaji for their continued support and encouragement.

I owe a lot to my friends too. Special thanks to Oguz Akyuz, who has been an awesome friend, philosopher and pal right from initial days at UCF. Discussions with him always brought the best out of me. My childhood friend Abhishek Srivastava always had encouraging words for me if I felt low during research. I also would like to thank the members of the NetSec lab for brainstorming discussions. Finally, I would like to thank all my friends I met along the way, Siddharth Borikar, Amit Shetti, Raju Sen Sharma, Prashant Goswami, Anirban Bag, Kiran Anna, Murat Balci, Kaushik Shirhatti, Dhanu Agnihotri, Amol and Shweta Marathe. I also wish to thank all those individuals who have been directly or indirectly related or helped me towards my PhD.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In last decade, Peer-to-Peer (P2P) networks have evolved from ground up. They have been extensively used for music sharing and now for sharing videos over the internet. The increasing deployment of broadband further revolutionized P2P media streaming applications which includes Video-on-Demand (VoD) and live media streaming. Live event broadcast [CRS02] using P2P technology such as Sports and TV streaming [Ppl05, Fei05] are more common now. Recent measurement studies [HLL06, AMZ06] have shown that users in excess of tens of thousands are turning to live streaming of popular Asian channels as of 2006.

P2P file sharing has been immensely popular because it does not necessitate a real time media playback and hence Quality of Service (QoS) and reliability issues are not a major concern. On the contrary, synchronous applications like live media streaming and VoD are characterized by strict time and bandwidth requirements, and have been partly successful. Moreover, the heterogeneity of peer bandwidths, peer location and topology, congestion in the interior of the network and lack of dedicated service has led to low reliability of P2P streaming. Researchers have applied various different paradigms to solve this problems but certain issues like QoS and reliability of service have never been near optimal. Recent measurement studies [HLL06, AMZ06] have presented new findings and problems about present day P2P media streaming systems.

1.1 P2P Media Streaming

Previous works in P2P media streaming focussed on improving certain aspects of media streaming metrics like QoS, bandwidth throughput, robustness and scalability using various paradigms of P2P streaming. However, these works lacked a collective evaluation and comparison of these metrics and their interdependence, which forms the underpinnings of an efficient media streaming system. Design flaws and inefficient peering strategy in the earlier works has led to the development of newer P2P streaming models, most of which are built on chunk-driven and *loosely coupled* peering philosophy.

1.1.1 Issues in Present day P2P Streaming

Users in excess of tens of thousands are turning to live streaming of popular Asian channels as of 2006 [HLL06]. Though it has been shown that P2P has emerged as a successful medium for live streaming, the quality of service (QoS) and reliability of streaming service still needs additional improvement. Recent measurement studies have revealed the following shortcomings of current day P2P streaming systems:

1. Recent study [HLL06] on PPLive showed that startup time of video before playback is in order of tens of seconds and sometimes even minutes, and needs to be minimized for a better viewing experience. The study further states that some nodes lag in their playback time by minutes as compared to their peers. We believe that this could be alleviated by a better peering strategy.

2. Another measurement study [AMZ06] on PPLive [Ppl05] and SOPCast [sop04] has shown that these streaming services lack *tit-for-tat* fairness that leads to uneven distribution of uplink bandwidth among users. The study found that these approaches use greedy algorithms for peering without the consideration of peer locality that leads to huge cross ISP traffic. An important finding of this work is that due to random data distribution structures, the uplink bandwidth utilization is sub optimal.

1.1.2 Motivation

The above limitations serve as a motivation for our current work. We propose a novel swarm based P2P model for live media streaming based on chunk-driven P2P philosophy. Our work mainly focuses on leveraging the randomness of swarm like environments and imposing a few management policies at the node level to reduce the real time packet contention among the nodes. The peering strategy and internal policies in our model are unique as compared to earlier works.

We propose a novel concept of alliance formation, where nodes cluster in groups, called *alliances*, for mutual node benefit while sharing the media content.

1.1.3 Our Proposed Methodology

We quantify QoS (in terms of jitterless transmission and latency), uplink bandwidth utilization, fairness (in terms of content served), robustness, reliability and scalability of the system, and design a suitable model to improve these metrics. We thus provide a comprehensive framework for collective evaluation of these media streaming metrics. We evaluate the effectiveness of our model and provide a comparative performance evaluation with DONet/CoolStreaming [ZLL05] system. We chose to compare our model with CoolStreaming for the following reasons: 1) It is based on swarming technology and uses chunk-driven P2P streaming philosophy. 2) It is the most important published work in recent times and can serve as a benchmark; many derivatives have evolved out of it that are extremely popular among audiences.

Further, we show that the node topology in our model forms a small world network [WS98]. We present an empirical analysis of our model under varying workloads and conditions. Results show that *alliance formation* is an effective way of organizing peers and distributing content in the P2P overlay networks. We show that our model has scaled well while achieving near optimal levels of QoS. We call our model as BEAM (Bit strEAMing).

Our proposed work focuses on architectural and organizational aspects of P2P media streaming. We do not deal with the media content and its various attributes like compression, audio/video quality and media format types. Different media formats vary in their space requirements. There is an on going research in making storage efficient and patent free formats for streaming by an end user. It is beyond the scope of our current work.

1.2 P2P File Sharing Domain: BitTorrent

The second part of the dissertation focuses on the current day challenges in P2P file sharing domain. We chose BitTorrent for our studies because it is the most widely used file sharing protocol as of today and largely accounts for P2P traffic over current day internet.

1.2.1 Current Issues in BitTorrent

Many studies in academia and industry have shown that though BitTorrent scales very well but is not near optimal in terms of fairness to end users, download time and uplink utilization. Furthermore, random peering and data distribution in such model lead to sub-optimal performance. Lately, new breed of BitTorrent clients like BitTyrant have shown successful strategic attacks against BitTorrent. Strategic peers configure the BitTorrent client software such that for very less or no contribution, they can obtain good download speeds. Such strategic nodes exploit the altruism in the swarm and consume resources at the expense of other honest nodes and create an unfair swarm. More unfairness is generated in the swarm with the presence of heterogeneous bandwidth nodes.

1.2.2 Our Proposed Methodology

We investigate and propose a new token-based anti-strategic policy that could be used in BitTorrent to minimize the free-riding by strategic clients. We also proposed other policies against strategic attacks that include using a smart tracker that denies the request of strategic clients for peer list multiple times, and black listing the non-behaving nodes that do not follow the protocol policies. These policies help to stop the strategic behavior of peers to a large extent and improve overall system performance. We also quantify and validate the benefits of using bandwidth peer matching policy. Our simulations results show that with the above proposed changes, uplink utilization and mean download time in BitTorrent network improves considerably. It leaves strategic clients with little or no incentive to behave greedily. This reduces free riding and creates fairer swarm with very little computational overhead. Finally, we show that our model is self healing model where user behavior changes from selfish to altruistic in the presence of the aforementioned policies.

1.2.3 Outline of the Dissertation

Related work is presented in chapter 2. In chapter 6, we present the preferential peering technique using strata based classification. We describe the details of our P2P streaming protocol in chapter 3. In chapter 5, we present details of how current day streaming ISP traffic can be brought down by using alliance based preferential peering. Chapter 7 present our phase II work for securing BitTorrent network from strategic threats. Finally, we present our conclusions and future work in chapter 8. We conclude the paper with the discussion of advantage of our work and future research directions.

CHAPTER 2

RELATED WORK

In this chapter, a taxonomy of P2P streaming methodologies is presented and some popular system in academia and industry are discussed briefly. Mainly, there have been two kinds of major approaches to solve the problems: Application layer multicast and P2P based models.

2.1 Taxonomy of P2P Media Streaming

2.1.1 Application Layer Multicast

After the success of P2P file sharing systems, many studies focussed on using the P2P technology on overlay networks for distributing the media content for viewing in realtime. Unlike Content Distribution Networks (CDN) like Akamai, P2P models were based on the principle of no infrastructure cost. The only resource an end user should contribute is its uplink bandwidth for itself and other members of the pool. The concept of IP multicast couldn't succeed for various technical, administrative and economic reasons. Application layer multicast (ALM) came up as an alternative

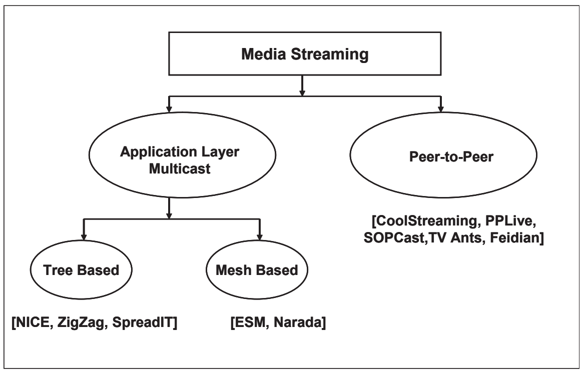


Figure 2.1: *Classification of P2P streaming Models*

to IP multicast, wherein multicasting functionality is implemented at end hosts instead of network routers. Application layer multicast builds an overlay network of participating users through unicast channels among the nodes. In ALM, most overlay network construction algorithms form a tree like node topology.

2.1.1.1 Tree Based

In tree based models, participating peers are organized into a single tree-structured overlay over which the media content is pushed from source towards all the peers. Some noted approaches in tree like structures are NICE [BBK02], ZIGZAG [THD03], and SpreadIT [DBG]. These approaches distributively construct an overlay network of nodes and routing functionality to minimize the number of hops for content distribution. The internal nodes in the tree are responsible for forwarding the content and any failure in these nodes causes short term failures including jitters in that particular sub tree before any repair algorithm can be used for recovery. An extension to this approach uses Multiple description coding (MDC) in which participating peers are organized into multiple diverse trees. Each description of the coded stream is sent through different subtrees. CoopNet [PS02] and SplitStream [CDK03] use MDC in their approach. CoopNet uses a centralized algorithm to obtain node information for tree construction, maintenance and data routing. SplitStream is a scalable tree based system built upon distributed hash table (DHT) based overlay network called Pastry. It strips the media content across a forest of interior-node-disjoint multicast trees that distribute the forwarding load among the participating nodes.

Advantages: Tree based model were comparatively easy to model. In a steady state, the data could be passed to child nodes in a regular pattern that brings down the latency [BBK02].

Disadvantages: Such models are susceptible to churn (frequent arrival and departure of nodes). Available bandwidth to each node in a tree structure is limited. Heterogeneous and asymmetric access link among peer nodes generates bottleneck, and leads to suboptimal uplink throughput and QoS is affected at user end [THD03].

2.1.1.2 Mesh Based

In mesh based models, participating peers are connected to many other neighbors in the pool like a mesh, in additions to the tree like topology. Narada and End System Multicast (ESM) [CRS02] are mesh based tree approaches to counter the problems in tree like structures. It has the capability to send multiple video streams at different qualities to counter node failures and degrading network links. It was also the first deployed system for broadcasting video streaming using ALM technology.

Advantages: They are less vulnerable as compared to tree based models owing to links to another nodes in the multicast trees [CRS02].

Disadvantages: The core topology of the network is tree like and susceptible to high churn. A node in a mesh based model keeps connections with many other nodes and it has comparatively high control overhead [CRS02].

2.1.2 P2P Based Models

Design glitches in P2P application layer multicast saw new generation of P2P models using chunk-driven P2P technology. In chunk-driven P2P approach, a file is typically broken into several small sized chunks of equal size and disseminated in the swarm. These models were heavily inspired by BitTorrent (BT) [Coh03] technology but their internal policies are very different from BT. Media content is broken down into small pieces and disseminated in the swarm. Participating peers form no strict topology like ALM but rather are randomly organized and connected to each other in mesh like models. The nodes periodically exchange the buffer information with their neighbors and trade unavailable pieces of content. It overcame the problems in tree based models and proved to be more robust in nature. Currently, these models are extremely popular and have a large viewer based. These P2P based models were applied to VoD as well as for live event streaming including TV broadcast.

2.1.2.1 Video-on-Demand Models

In VoD, media content is streamed to an end system after a user makes a request for it. Participating peer in the swarm help each other with their uplink bandwidth. Different viewers in the swarm may view different part of the video since they requested the media at different times. BASS [CC05] is a recent P2P streaming technique for VoD thats uses a hybrid approach of BT and a client-server model providing the swarm with an external media server. However, load on such server increases linearly with the number of users due to its server centric design and hence does not scale well. BiToS [VIF06] is a BT modified approach to VoD using the P2P network. Redcarpet [SM05] is yet another work that concentrates on providing near VoD using different piece selection algorithms in BT. Since BT has been proven to be near optimal in achieving uplink utilization and mean download time, many models have modified BT to suit the VoD needs.

2.1.2.2 Live Media Streaming Models

Live media streaming is event based and streaming is broadcasted in real time, with all the viewer viewing the same content with very little differences in their delays. PRIME [MSR05] is a mesh based P2P streaming approach to live media streaming that focuses on finding the global content delivery pattern to maximize the uplink utilization while maintaining the delivered quality. PROMISE [HHB03] is a system that uses an application level P2P service called Collectcast for peer selection and dynamic reconfiguration in case of sudden network failures and topology changes. CoolStreaming [ZLL05] is one of the most successful P2P approaches to live media streaming. It is based on a data driven overlay network where a node periodically exchanges data availability information with a set of partner and retrieves the unavailable data and helps peers with deficient content. A new breed of piece based P2P system have evolved since CoolStreaming's immediate success. Proprietary models like PPLive, SOPCast and TVAnts are derivatives of it but their exact working philosophy is unpublished. Piece based P2P model exploits the randomness of the swarming technology (very similar to BitTorrent (BT)), and doesn't depend on tree based structures. Server relays the content in small pieces and disseminates in the swarm. Every node typically has 6-8 neighbors and they exchange information about their buffer and then trade the unavailable pieces.

Advantages: Pure P2P models are more robust, scalable as compared to ALM models. Random swarm provides multiple channels to provide data in shortest possible time. CoolStreaming like deployed systems have shown the feasibility of such approach [ZLL05].

Disadvantages: Control overhead is relatively more than ALM models. Random nature of swarm brings in more startup time before media playback begins as there is no fixed pattern of content delivery. Other issues like fairness and excessive cross IP traffic have surfaced recently [ZLL05].

CHAPTER 3

BEAM: OUR PROPOSED METHODOLOGY

BEAM [PG07a, PG07b] consists of three main entities: nodes, a media relaying server and a tracker. Media relaying server is the origin of the stream content in the swarm. The tracker is a server that assists nodes in the swarm to communicate with other peers[1]. It also communicates with the media relaying server to exchange important information about the current state of the system. As a new user arrives, it contacts the tracker and submits its IP address together with its bandwidth range. The tracker issues it a peer list, typically 40 nodes, from the set of nodes that are in similar bandwidth range. Alternatively, if it is not available, tracker provides the list of nodes in the closest bandwidth range. Small et al. [SLL06] and Bharambe et al. [BHP06] have shown that interaction of nodes in similar bandwidth range leads to optimal resources utilization in the swarm. The new node requests stream content from the nodes in its peer list, and then starts creating and joining alliances. Alliance formation is explained in detail in Section 3.1.

Since the media relaying server cannot stream the content to multiple users simultaneously due to the bottleneck in its uplink speed, it streams the content to a selected number of peers, termed as *power nodes*, which have higher contribution to the swarm in terms of content served. Initially,

[1]Nodes and peers have been used interchangeably.

when the streaming starts, power nodes are chosen from the nodes with higher uplink bandwidth, since the contribution of nodes is yet undetermined. The power nodes in turn forward the content to the other peers in the swarm.

The tracker periodically (e.g. every 10 minutes) computes the rank of the nodes in terms of the content served to the swarm. If the media server can simultaneously stream the content to, say P nodes, then the P top ranked nodes become the power nodes. The tracker updates the media server about the new power nodes, which are then streamed the media content directly from the server. The rank is calculated on the basis of a *Utility Factor* (UF), which is a measure of the node utility or contribution to the swarm. UF is computed using two parameters: *Cumulative Share Ratio* (CSR) and *Temporal Share Ratio* (TSR). Share ratio is the ratio of the uploaded volume content to the downloaded volume content by an end user. CSR is the share ratio of a node since its arrival in the swarm, whereas TSR is the share ratio over a recent period of time. Thus, $UF = f(TSR, CSR)$. We formulate UF as follows:

$$UF = \alpha CSR + (1 - \alpha) TSR$$

where α is the weight of CSR and $(1 - \alpha)$ is the weight of TSR. For example, if a node has a $CSR = 2.0$, $TSR = 4.0$ and $\alpha = 0.75$, then $UF = 2.5$. Only the nodes that have (CSR, TSR) values ≥ 2.0 (empirically obtained from Figure 4.3(a) and explained later) periodically update the tracker with their (CSR,CSR). These account for less than 20% of the total nodes in the swarm (see Figure 4.3(a)). These 20% nodes are enough to generate the required number of power nodes in a streaming session and do not incur significant overhead since the remaining 80% nodes do not report to the tracker. This alleviates the tracker from receiving an overwhelming number of messages from the nodes in the system. We assume that nodes are honest and do not tamper with the data, protocol and the software at the client end. Similar concept of the gauging share ratio of registered users is used in popular BitTorrent clients like Azureus [Azu].

Since the power nodes are periodically computed based on their UF, they need to perform consistently well in terms of distributing the content to remain as power nodes, else they could be replaced by other well performing nodes. The purpose is two fold: 1) It serves as a natural incentive for the power nodes as well as the non power nodes to contribute to the swarm since this reward helps them to get the content early and directly from the server; the most reliable source in the swarm. Such altruism has been shown to be very effective in improving the overall swarm performance [PIA07, BHP06]. 2) Nodes with higher uploading capacity are closer to the server. Small et al. [SLL06] has proven that placing peers with higher uploading capacity closer to the source achieves optimal performance in terms of maximizing uplink utilization and minimizing average delay for all the peers in the swarm.

Live media streaming is time and resource constrained. Nodes contend within themselves for the same media content within a short period of time. The need to playback the media and procure the future content necessitates an effective management policy. We introduce the concept of *alliance formation* to counter these problems. Nodes cluster into small groups, typically between 4 to 8, called *alliances*, to form a symbiotic association with other peers. Members of an alliance are assumed to be mutually trusted and help each other with sharing media content. Our model places an upper bound on two very important parameters: Maximum number of nodes in an alliance, h, and maximum number of alliances a node can join, k. A node can be a member of at most k alliances and this helps the node to form a stronger connectivity in the pool and gives an option to receive the stream content from different alliances (paths).

As a power node receives the media content from the server, it propagates the content within its alliances. While serving the content to its alliance members, a node serves different pieces of a packet[2] to its peers. A packet contains $(h-1)$ pieces, which the power node distributes to the other $(h-1)$ members of the alliance, i.e. each node gets one piece, which it shares with other alliance members and subsequently obtains other missing pieces from other members. In this process, a node downloads $(h-1)$ pieces and uploads $(h-2)$ pieces. This is done to leverage the uplink bandwidth of all the peers and make participation necessary so that no node gets a free ride. In case a node cannot get a particular piece because it could not fetch from peer in its alliance, it can request the power node for it. Nodes that only procure the content from alliance members and do not share are ignored by other alliance members in future and alliance member find another node to replace such non contributing node to be in the alliance. As a node gathers all the pieces of a packet, it starts the media playback, forwards the content among its other alliances like power node and procures the future stream content. A node uses *announce mechanism* to notify its alliance members the receipt of a new packet. This process of announcing and exchanging unavailable content can also be efficiently improved using Network Coding [GR05]. Periodically, nodes in an alliance serve a request that is out of the alliance to bootstrap a new node.

[2]A packet refers to a collection of pieces and does not refer to an IP packet. A piece is the smallest data unit exchanged.

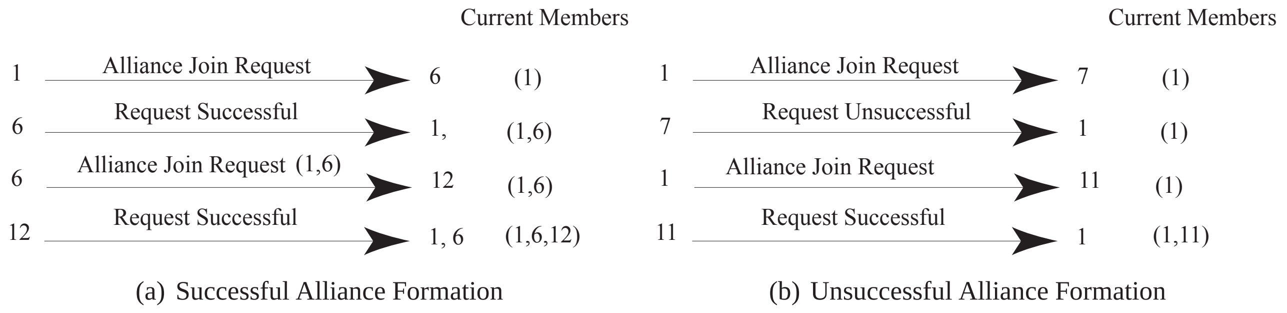


Figure 3.1: *Alliance Formation in BEAM*

3.1 Alliance Formation

A node creates an alliance by sending an alliance join request packet to the nodes in its peer list. The receiving node can accept the alliance join request or reject it (depending on how many alliances it is currently a member of i.e. k). In case of rejection or no reply, node times out after a short time interval (e.g. 2 round trip times (RTT)) and continues to search for peers to join their alliances. If a node accepts the alliance request, it issues a success packet back to the requesting node. These two members of the alliance can expand and grow the alliance further. The format of a request packet for an alliance is shown by: $[A_{ID}, Num, N_1, N_2, ..]$, where A_{ID} is the ID of the alliance, Num denotes number of current members in the alliance, and N_{id} is the ID of the present member(s) in the alliance. N_1 is the sender of the alliance join request. The format of a success packet is as follows: $[A_{ID},\ Self_{ID}]$, where $Self_{ID}$ is the ID of the node that sends the success message to all the alliance members in A_{ID}. Figure 3.1 depicts the process of alliance formation. In Figure 3.1(a), following events occur:

1) Node 1 sends an alliance request to node 6.

2) Node 6 accepts alliance invitation and returns success packet to node 1.

3) Node 6 issues a request packet to node 12, that includes: alliance ID and IDs of nodes 1 and 6.

4) If 12 joins the alliance, 12 will send success packets to both, 1 and 6. Now all three nodes 1, 6 and 12 are members of the same alliance. Nodes expand the alliance till k is reached.

Similarly, in figure 3.1(b) following events occur:

1) Node 1 issues an alliance request to node 7.

2) Node 7 does not reply or rejects the request. This could be because node 7 has reached the maximum limit of k. Node 1 times out after a small time interval.

3) Node 1 issues request to some other node, say 11.

4) If 11 agrees to be part of the alliance, it sends success packet to node 1. Nodes 1 and 11 are members of the same alliance.

Nodes expand the alliance till k is reached. Once k is reached and if a new node requests to be a member of the alliance, it cannot be included. The other node in such a case can initiate the formation of a new alliance. One important point to note is that even if a node is not a member of any alliance it request nodes in its peer list to join its alliances. It depends on the other node whether to join the alliance or not.

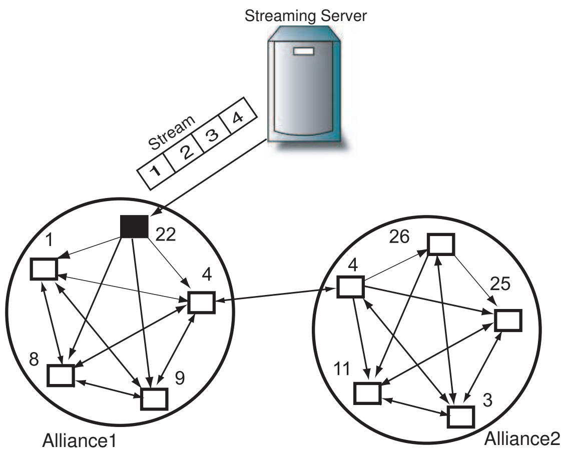


Figure 3.2: *Alliance Functionality*

3.2 Alliance Functionality

A node can be member of multiple alliances (at most k). This is important to facilitate multiple paths for a node to obtain the stream content in case of node failures. As a member of an alliance procures a packet, it spreads it among its respective alliances. Consider the scenario in Figure 3.2, Alliance1 consists of nodes with IDs (1, 4, 8, 9, 22) and Alliance2 has nodes with IDs: (3, 4, 11, 25, 26) with node 4 being a member of both the alliances. Suppose, node 22 obtains a new packet from one of its other alliances or from media server, it then forwards it in Alliance1. It sends an announce packet to its members as: $[A_{ID}, PNum, NPieces]$, where $PNum$ is the packet number in the streaming and $NPieces$ is the number of pieces in the packet. Nodes (1, 4, 8, 9) request for unavailable pieces that they need to procure to complete the download by sending a request in the form: $[A_{ID}, PNum, P_1, P_2, ...]$, where P_1 and P_2 are piece number 1 and 2 respectively.

A packet comprises of $(h-1)$ pieces. If all the members of a particular alliance simultaneously request all the pieces, the forwarding node randomly distributes the pieces of the requested packet among them. It is left to the peers to exchange the pieces within themselves. In case, a node requests specific unavailable pieces in a packet since it has already obtained some pieces from other alliances, the forwarding node sends only those specific requested pieces to avoid any redundancy at the requesting node. In the above example, if members of Alliance1 have procured distinct pieces, they exchange among themselves to complete their individual downloads. As nodes of Alliance1 procure the complete packet, they forward it in their other alliances. In the above case, node 4 (common node in both the alliances) forwards the content in Alliance2 by announcing the arrival of the packet and the subsequent process of forwarding the content is similar as explained above.

While leaving the network, a node sends a departure packet to its alliance members as follows: $[A_{ID}, Self_{ID}, Flag_D]$, where $Flag_D$ is the departure flag. In case a node exits without sending a departure packet, the nodes within the alliance become aware of its inactivity and infer its departure. Other nodes in the alliance continue sharing the streaming content within themselves and/or can find another member for the alliance. In our model, we propose to use TCP connection in BEAM as the network links between peers. TCP detects the peer's departure from the pool and gracefully handles shutdown of the connection link. Li [Li04] has explained the benefits of using TCP over UDP/RTP, provided the initial buffer time is significantly larger than the Round Trip Time (RTT) between the peers. We elaborate the details in the simulation section.

3.3 Small World Network

In this section, we present an analogy between BEAM and Small World Network (SWN) [WS98] to show the effectiveness of BEAM's important properties such as near-optimal overlay distance and network robustness in the events of churn and node failures. SWN is a class of random graphs where: 1) Every node has dense local clustering, i.e a high coefficient of clustering (μ_c, defined below) and some edges with far located nodes. 2) Every node can be reached from every other node by a small number of hops or steps. We present a graph theoretic analysis of our model and show that it generates a swarm of nodes, which when converted to a graph, end users as vertices and connection between them as edges, exhibits small world network characteristics. We compare our results with CS [ZLL05] which uses a random network topology. Random graphs [Bol01] are known to generate near-shortest mean distance between all pairs of nodes in a graph.

We chose to show an analogy with small world network for the following reasons: 1) Overlay hops (path length) between any two nodes is short in SWN and partially reflects end to end latency [BBK02, THD03, ZLL05]. 2) High local clustering means a close knit group ; in a media streaming scenario it ensures that once a packet is in the alliance, it can be readily obtained from the alliance members. The important group policies required in an alliance can also be readily applied. 3) SWN are robust to network perturbations like churn and hence provide an efficient overlay structure in events of nodes failures.

3.3.1 Alliances and Small World Network

μ_c is a local property of a vertex v in a graph and is defined as follows. Consider the vertex v and a set of neighboring vertices V = (v_1, v_2, $\ldots$,v_n) and a set of edges E, where e_{ij} denotes an edge between the vertex i and vertex j. The clustering coefficient(μ_c) of a vertex is the ratio of actual number of edges present to the total possible edges among those vertices:

$$\mu_c = \frac{|e_{ij}|}{\binom{n}{2}} = \frac{2|e_{ij}|}{n(n-1)}$$

In other words, μ_c is density of edges in the node's vicinity. Average of clustering coefficients of all the nodes is the clustering coefficient of the graph. Mean path length is the mean of path lengths between all pairs of vertices in the graph. The concept of SWN is counter intuitive as graphs with higher clustering coefficients would be dense locally and require more hops to traverse the other parts of the graph as compared to a random graph. Watts et al. [WS98] showed that routing distance in a graph is small if each node has edges with its neighbors (i.e. has high μ_c) as well as some randomly chosen nodes in the swarm. Similarly, Kleinberg [Kle00] proved that if every node in the swarm shares an edge with a far located node, the number of expected hops for routing between any pair of vertices becomes $O(log^2 N)$, where N is the number of nodes.

Suppose a node is a member of k alliances (a_1,a_2,.....a_k) and each alliance has neighbors (m_1, m_2, ...,m_k), where $|m_i| \leq h$, and $1 \leq i \leq k$. Therefore, coefficient of clustering for such node would be:

$$\mu_c \geq \frac{\binom{m_1}{2} + \binom{m_2}{2} + ... + \binom{m_k}{2}}{\binom{m_1+m_2+m_3....+m_k}{2}}.$$

Figure 3.2 depicts the neighborhood of node 4. It is a member of two alliances and in each alliance, it is connected to four other members. Nodes in an alliance forms a clique (complete subgraph). Node 4 is completely connected to members of Alliance1 and Alliance2, though, members of Alliance1 and Alliance2 may or may not be connected with each other. With respect to Alliance1, node 4 forms four long distance links elsewhere in the network. Similarly, with respect to Alliance2, node 4 forms four long distance links elsewhere in the network. This property is analogous to small-world network, where nodes are well connected locally and also have some long distance links elsewhere in the network that helps to achieve a small path length between all pairs of nodes. Coefficient of clustering for node 4 in this case would be:

$$\mu_c \geq \frac{\binom{4}{2} + \binom{4}{2}}{\binom{8}{2}} = \frac{3}{7} = 0.428$$

We also consider the case that other alliance members can have edges between them. Similarly, other nodes in the graph would have μ_c of at least 0.428 since they have similar constitution of alliance and neighborhood. Clustering coefficient of 0.428 is relatively much higher than a random graph (μ_c for random graph of the same size was found to be 0.0019), and therefore it lies in the region of small world graphs as mentioned in [WS98].

3.3.2 Graph Theoretic Properties of Alliance

Graph density (ratio of number of edges to the total number of possible edges in the graph) is an important factor for the connectedness of a graph. We evaluate the graph density of a BEAM graph by abstracting the alliances as nodes. As a member of an alliance receives a packet, it forwards within its alliance members and hence we focus on alliance hops rather than individual node hops in this scenario and compute the same. To simplify, we consider an alliance as a single node which we call super node. Suppose there are N nodes in the swarm viz (V_1, V_2,. ....,V_N) and they are spread in M alliances. Let D_{graph} be the density of the graph, $D_{alliance}$ be the density of the graph when alliances are abstracted as vertices i.e. super nodes as vertices, M be the number of super nodes in the swarm, O be the outdegree of a super node. Every node in the swarm is connected to $(h-1)$ other nodes in every alliance and there are k such alliances. Therefore, we have

$$D_{graph} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{k}(h_{ij}-1)}{2*\binom{N}{2}} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{k}(h_{ij}-1)}{N*(N-1)}$$

where h_{ij} is the number of members in j^{th} alliance of node i, and $1 \leq i \leq N$, $1 \leq j \leq k$. In a steady state, when all the nodes have formed k alliances, and each alliance has exactly h members, we have

$$D_{graph} = \frac{(h-1)k}{N-1}$$

Since super nodes are formed by contracting the alliances, we have

$$M = \frac{N * k}{h}$$

Every super node is connected to other super nodes through its h members and their respective $(k-1)$ alliances since every node is a member of k alliances each. Therefore, we have

$$O = h * (k-1)$$

Since there are M super nodes and each has a outdegree of O, there are $\dfrac{M * O}{2}$ edges. Therefore, we have,

$$D_{alliance} = \frac{\frac{M*O}{2}}{\binom{M}{2}} = \frac{h^2(k-1)}{(Nk-h)}.$$

For $h = 5$, $k = 2$, i.e. node degree = $(h-1) * k$ = 8 and $N = 512$, the D_{graph} is approximately 0.004, while $D_{allaince}$ is approximately 0.025. We see that the density of the graph at alliance level is relatively much higher than at the node level i.e. $D_{alliance} >> D_{graph}$. The alliance formation and subsequently the topology of the network produces strongly connected graph and reduces the hop count during the communication. Similar abstraction is not possible for complete random graphs.

We are more interested in the mean path lengths from server to nodes rather than mean path lengths between all pairs of nodes. Therefore, we limit our search to length of all the paths from server to all other nodes in the swarm.

Consider, a tree like view of BEAM graph. Note that a tree like view is a simplification of BEAM network since in BEAM, many nodes and hence alliances are joined with each other and it is a mesh of nodes rather than a tree like topology. We depict it as a tree like structure to quantify its path length from the source server. Consider L_1 as path length in a conventional tree and L_2 as path length in BEAM like graph.

$$L_1 \leq \log_{NodeDegree} N$$

$$L_2 \leq \log_{k(h-1)} N$$

It is trivial to see that in conventional tree like topologies, the path length from source to any node is bounded by L_1 . The hop count in a BEAM network is bounded by L_2, since a node after procuring the content forwards it in its other k alliances i.e. to $k(h-1)$ nodes. Since, BEAM graph has lot of interconnections between them, the above equations only depict the upper bound. It is more difficult to infer the same on random graphs. To gauge the actual path length in a large swarm, we conducted experiments for finding average path length between all nodes, average path length from server, radius, and diameter of a graph.

Table 1 illustrates results from a simulation in which we compare synthetically generated random graphs with BEAM graph having the same node degree and overall density in the graph. We

Table 3.1: *A comparison of BEAM, Random and a network generated graph. The experiment was conducted for 512 nodes, node degree 8.*

Graph Type	Diameter	Radius	Mean Distance	Server Distance	Clustering Coefficient
BEAM	6	5	3.37	3.19	0.42
Hybrid	5	4	3.33	2.87	0.014
Random	5	4	3.26	3.16	0.013

use networkx python library [net] for complex networks to obtain our simulation results. In these experiments, we have tested 3 kinds of graphs: completely random graph, BEAM network graph and hybrid BEAM graph where alliances acts as nodes. The graph density, node degree = 6 and node count = 512 were same in all the three graphs. Hybrid graph's node count is reduced to $\frac{Nk}{h} = 256$ since $h = 4$, $k = 2$, and degree of such hybrid nodes is equal to h. Server distance is calculated by picking a random node among the 512 nodes and then calculating distance from it. From the results in table 1, it is seen that random graphs perform well as expected in all the metrics. BEAM graphs have performed at par with random graph. Server distance in hybrid graphs is even shorter than random graphs. Random graphs have relatively lower mean path length but hybrid have lowest mean server distance. This abstraction of BEAM graphs helps to analyze the topography and various other graph theoretic properties.

Figure 3.3 depicts performance of random and BEAM graphs for higher number of nodes. These values were found by averaging 10 different runs using the networkx python library for complex graphs. Random graphs are known to perform better while traversing the graph. It is evident from the figure that it has relatively shorter radius and diameter as compared to BEAM graph. BEAM has nearly matched random graphs in mean distance from server, which is the most important criteria in our environment. Thus, BEAM graph forms a small world network with

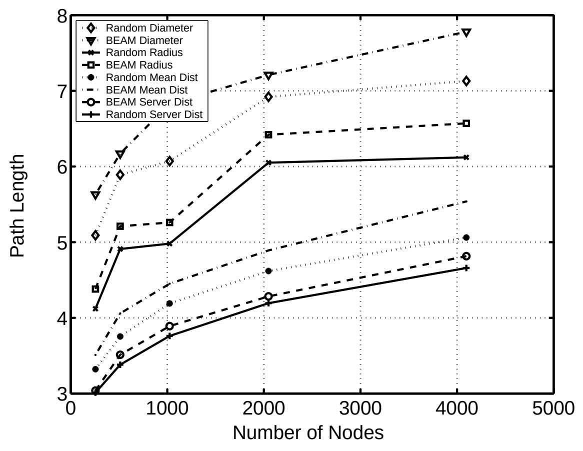


Figure 3.3: *Comparison of Random and BEAM graph*

relatively much higher clustering coefficient and very comparable mean path length to the random graphs.

3.4 Summary

In this chapter we introduced our P2P streaming framework, BEAM, that is based on alliance based peering scheme and forms a small world network. Alliance formation is an effective organization of peers into small groups where a node contributes effectively and also gets served by alliance members effectively. Small world network typically have short path lengths and are robust to network perturbations such as churn. As we have shown that alliance based network topology forms a small world network and displays short path length. The results in subsequent chapter demonstrate that it indeed forms a very robust overlay network and is very stable during churn and during other network anomalies.

CHAPTER 4

SIMULATION

In this chapter, we describe our simulation setup and discuss the results. Initially, we discuss the metrics which we have used in our simulation to compare our model with CoolStreaming.

4.1 Metrics

In this section, we define the metrics that are most important in a P2P streaming environment. We quantify these metrics and provide mathematical expressions for the same. In our simulation, we use the following expressions to evaluate our results.

Metric 1. Average Jitter Factor = $\left(\sum_{i=0}^{N} J_i\right) \Big/ N$ where,

$$J_i = \left(\sum_{i=0}^{T} F_i\right) \Big/ T$$

$$F_i = \begin{cases} 0 & \text{if packet arrived before media playback} \\ 1 & \text{if packet not arrived before media playback} \end{cases}$$

Here, T and N denote the total packets in the streaming session and the total number of nodes in the swarm respectively.

If a packet is not received at its playback time, it is considered to be a jitter. Average jitter factor is critical to maintain high quality of streaming at user end; lower the jitter rate, better is the QoS. Since, it is averaged out among the total number of end users, it depicts a system wide measure of QoS. We compute the jitter factor for each individual node and then average it to compute the system wide jitter factor. Jitter factor is also called as continuity index in some previous works [ZLL05].

Metric 2. Average Latency = $\left(\sum_{i=0}^{N} L_i\right) \Big/ N$ where,

$$L_i = T_{\text{Node}_i} - T_{\text{Server}}$$

$$T_{\text{Server}} = \text{Media playback time at Server}$$

$$T_{\text{Node}_i} = \text{Media playback time at Node}_i$$

The difference in media playback time at user end and server end is the latency. Most live events and their streaming rely on minimizing the latency from the actual playback at server end to keep end users interested. We compute the latency of individual nodes and then average it to derive the system wide measure of average latency.

Metric 3. Uplink Utilization = $\frac{\text{Total Uplink Used}}{\text{Total Uplink Available}}$

The better the uplink utilization, better is the scalability of the system. Uplink bandwidth is the most sparse resource in the swarm and its maximization leads to optimal performance in the swarm in terms of minimizing delay and maximizing number of end users [SLL06].

Metric 4. Fairness Factor = Variance$(SR_1, SR_2 \ldots, SR_N)$ where, SR_i denotes the share ratio of node i and is defined as,

$$SR_i = \frac{\text{Uploads } (\text{Node}_i)}{\text{Downloads } (\text{Node}_i)}$$

Fairness can be defined in several different ways, for e.g. in terms of uplink bandwidth contribution, pricing etc. We believe that in such random swarm environments, it is extremely difficult to deliver services in proportion to their contribution. We define fairness in terms of share ratio of content served by nodes. Share ratio of end users over the period of simulation run depicts the contribution of the nodes quite fairly. Since, it is difficult to provide services in proportion to the contribution, the best we can do is to minimize the variance of share ratios of the nodes in the swarm by enforcing strict policies. An ideal system would have nodes with share ratios of 1.0, where an end user gets streaming content and it passes on equally to other end user. But given the dynamics of the internet, it is very difficult to achieve the same. More the number of nodes close to share ratio of 1.0, the fairer is the system.

Metric 5. Robustness Factor = Maximum(F) where,

$$\begin{aligned} F &= \text{Percent failure of nodes} \\ R_J &= \text{Average Jitter Factor} \\ R_L &= \text{Average Latency} \\ \Delta R_j &= \text{Threshold Jitter Factor} \\ \Delta R_l &= \text{Threshold Average Latency} \end{aligned}$$

and such that,

$$R_J \leq \Delta R_j$$

$$R_L \leq \Delta R_l$$

To evaluate the robustness of BEAM with respect to achieving acceptable levels of QoS, while maximizing the node failure rate in the swarm, we assign a threshold of Δ_{Rj} and Δ_{Rl} to jitter factor and average latency respectively. We test the robustness and reliability of the underlying network architecture under increasing node count, subjecting the system to varying percentages of node failures or departures. We determine the maximum node failures which the system can withstand, without degrading the QoS.

Metric 6. Scalability Number = Maximum(N) where,

$$
\begin{aligned}
N &= \text{Number of Nodes} \\
S_J &= \text{Average Jitter Factor} \\
S_L &= \text{Average Latency} \\
\Delta S_j &= \text{Threshold Jitter Factor} \\
\Delta S_l &= \text{Threshold Average Latency}
\end{aligned}
$$

and such that,

$$
\begin{aligned}
S_J &\leq \Delta S_j \\
S_L &\leq \Delta S_l
\end{aligned}
$$

To evaluate the scalability of BEAM with respect to achieving acceptable levels of QoS while maximizing the number of nodes, we assign a threshold of Δ_{Sj} and Δ_{Sl} to jitter factor and average latency respectively. The scalability number indicates the optimal number of users in the swarm, where the threshold is not exceeded and number of users are maximized.

Table 4.1: *This table explains all the Metrics related terms. All these terms are part of some expression in the Metrics in Table 4.2.*

Term	**Expression**	**Description**
Total Number of Nodes	N	Number of Nodes in swarm
Total Number of Packets	T	Total Packet in a Streaming Session
Number of Server Upstream Connection	P	Server can simultaneously upload to P nodes
Total Uploads by node i	UP_i	Volume of Uploads by node i
Total Downloads by node i	$DOWN_i$	Volume of Downloads by node i
Share Ratio of node i	SR_i	$UP_i/DOWN_i$
Media Playback Time at Server	T_{server}	Start Time of Media Playback at Server
Media Playback at Node i	T_i	Start Time of Media Playback at Node i
Piece Availability	F_i	0 if packet available before media playback else 1
Jitter Factor of Node i	J_i	$\left(\sum_{i=0}^{T} F_i\right) \Big/ T$
Latency of node i	L_i	$L_i = T_{\text{Server}} - T_{\text{Node}_i}$

Table 4.2: *This Table lists all the Media Streaming Metrics we have used in our simulations. For explanation on the terms refer to Table 4.1*

Term	**Expression**	**Description**
Average Jitter Factor	A_j	$\left(\sum_{i=0}^{N} J_i\right) \Big/ N$
Average Latency	A_l	$\left(\sum_{i=0}^{N} L_i\right) \Big/ N$
Total Uplink Utilization	U	$Uplink\ Used/Uplink\ Available$

4.2 SIMULATION SETUP AND EXPERIMENTS

To evaluate various aspects of BEAM which are normally difficult to study using logs of real world torrents or trace, we use a simulation based approach to study the same. A simulator gives the ability to experiment with different parameters involved in the system and study its performance under varying workloads and conditions. Experiments for large scale simulation in excess of thousands of nodes are difficult to perform using real world implementation for lack of nodes participating in it. Similarly, such experiments have limited domain in testbed overlay networks like Planet Lab[Pla] due to the limited number of nodes[1] participating in it, while it can be suitably modeled in a simulator. It also helps to check the feasibility of such a system and to verify if simulation results corroborate with the analytical results. Though, it is difficult to capture all the internet dynamics correctly in a time event simulator, we mention the assumptions and simplifications we make, and how it will impact results in real world scenario. With these assumptions and simplifications, we are able to model and simulate the behavior of BEAM and CS faithfully and the results show the definitive trends and directions.

4.2.1 Simulator Details

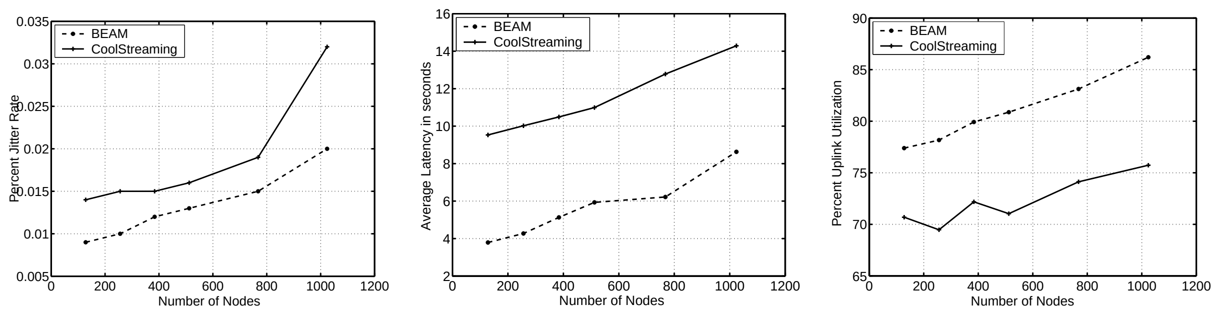
We simulate both the models i.e. BEAM as well as CS [ZLL05] and compare their results based on the metrics defined in Table 4.2. We simulate all the components of CS i.e. 1) Node join and

[1]As of October 2006, PlanetLab currently consists of 704 machines, hosted by 339 sites, spanning over 25 countries.

membership management algorithm. 2) Buffer map representation and exchange. 3) Intelligent scheduling algorithm. 4) Failure recovery and partnership refinement method. In BEAM, we model the server, tracker functionality, and the nodes in the swarm. Server is the only source of streaming packets in the system. For comparing the two systems, we quantify QoS (in terms of jitter factor and latency), uplink utilization, fairness (in terms of content served by an end user). We also analyze robustness, reliability and scalability of the system by evaluating the QoS of the system under varying workloads and conditions like node failure, churn, larger swarms etc.

We used the BRITE universal topology generator [MLM01] in the Top-Down Hierarchical mode to model the physical network topology of Autonomous Systems (AS) and the routers. All AS are assumed to be in the Transit-Stub manner. Overlay is assumed to be undirected. Unlike other simulators [BHP06, BCC06, MSR05], we assume that the bottleneck in the network can appear in the access links of source and destination (i.e. first-mile and last-mile hops) as well as the non access links that are in the interior of the network, in particular within or between carrier ISP networks. The nodes in the swarm are assumed to be of heterogeneous bandwidth classes namely:(512Kb, 128Kb), (768Kb, 256Kb), (1024Kb, 512Kb), (1536Kb, 768Kb), (2048Kb, 1024Kb) where first and second member of the tuple are the maximum downlink and uplink speed of a node respectively. The distribution of these bandwidth classes is uniform in the swarm. To simulate the congestion in the Internet, we induce 5% congestion in the non access links within the interior of the network. In such congestion scenarios, the available bandwidth to nodes is the minimum of the bottleneck at source or destination and the bottleneck in the non access links. The delay on inter-transit domains and intra-transit domains are assumed to be 100 ms and 50 ms respectively, while delay on stub-transit is assumed to be 30 ms and intra-stub transit links are randomly chosen between 5ms and 25ms. We simulate the TCP level dynamics like timeouts, slow start, fast recovery and fast retransmission by introducing a delay of 10 RTTs [CK01]. We model a flash crowd scenario for the arrival of users in the swarm, i.e. all users are present in the swarm when the live media streaming starts, as this is the most relevant and challenging scenario for the P2P streaming system.

In our experiment, the number of nodes typically vary from 128 to 4096 for most cases. For some large sets of experiments we have also considered nodes in excess of 16000. We consider a media file of duration 120 minutes, originating from a source, encoded with streaming rate of 512 Kbps and a file size of approximately 440 MB. In BEAM, we use the values of $(h, k) = (4, 2)$ to make the neighbor count $= 6$, similar to CS, for a fair comparison. Table 4.3 provides other values of (h, k) that can be considered for streaming. The values of α is 0.75 from Table 4.4. Each piece size is 64 Kb and hence packet size is $(h - 1) * 64$ Kb = 192 Kb in our case. We maintain similar settings for the remainder of the paper. Any changes in the configuration settings are mentioned in the respective sections.



(a) % Jitter Rate vs Number of Nodes (b) Average Latency vs Number of Nodes (c) % Uplink Utilization vs Number of Nodes

Figure 4.1: *Comparison of QoS parameters in BEAM and CoolStreaming*

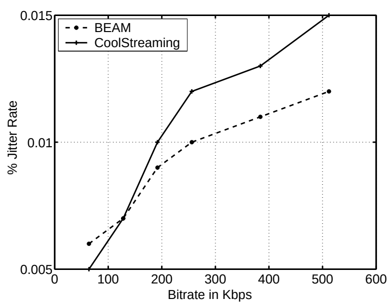
4.2.2 Results and Discussion

4.2.2.1 QoS and Uplink Utilization

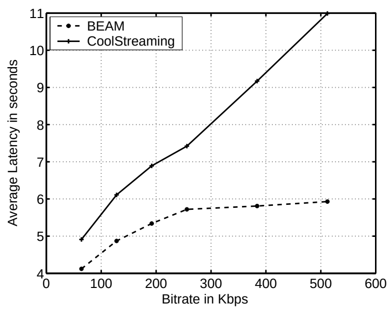
In the first set of experiments, we compare the effectiveness of alliance theory of BEAM on QoS and uplink utilization as against CS's random peer selection. Figure 4.1 depict these comparisons. In accordance with the conventional notion of scalability in P2P systems, it can be seen from figure 4.1(a) that BEAM and CS both perform better with the increasing swarm size, though jitter rate slightly increases after 1500 node mark but stabilizes around 2000 nodes. BEAM has a comparatively lower (approximately 0.01%) jitter rate than CS. The plausible reason is that in CS, the content delivery is random in nature rather than an organized flow. Sometimes an intermediate piece which could not be fetched, may increase the jitter rate. Due to alliance formation in BEAM, the stream content propagates in an organized fashion from one alliance to other; so chances of an intermediate piece missing are comparatively low. In BEAM, every node receives the content through the best possible channel among its various alliances, while the same cannot be commented for CS. An optimal jitter rate of 0 is difficult to attain in such random swarm environments because the content distribution is dynamic and, lasts for an extended time; network anomalies and congestions can cause unavailability of a packet at its playtime.

Figure 4.1(b) depicts the average latency in both systems. For the same settings, average latency for BEAM varies from less than 10 seconds for a 128 node swarm to less than 18 seconds for a 4096 node swarm, while CS has considerably higher latency of 29 seconds for a swarm of 4096 nodes. An explanation for this could be that higher the number of hops in the overlay, greater are the chances of increased end to end latency [BBK02, THD03]. CS has comparatively higher bootstrapping time before playing media and it could be attributed to the following facts: 1) It buffers more pieces in advance before playback. 2) Due to random nature of data exchange, a missing intermediate piece further increases jitter and hence latency. 3) Execution of intelligent scheduling algorithm causes both computation overhead and delay. On the contrary, in BEAM, the systematic flow of content from one alliance to another and near optimal overlay hops account for its lower latency. Moreover, if a packet has been procured by an alliance member, it implies that there are at least one or more sources for the content. This flow of packets indeed saves time as compared to CS. Playback starts 10 seconds after receiving the first segment in [ZLL05]. In our implementation of both BEAM and CS, the playback starts after 6 seconds, as 6 seconds of buffer time is long enough and is many times larger than the RTT between peers to counter the network anomalies like jitter and congestion within the network [Li04].

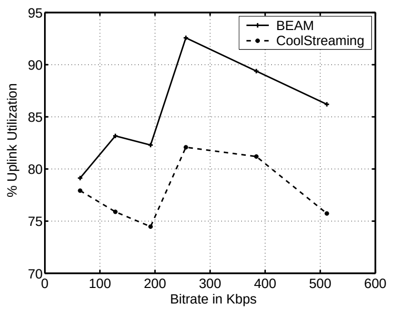
Uplink bandwidth is the most important resource of a P2P system. End users that are charged for bandwidth used per time unit want to maximize their utilization. Moreover, maximization of



(a) % Jitter Rate vs Bitrate



(b) Average Latency vs Bitrate



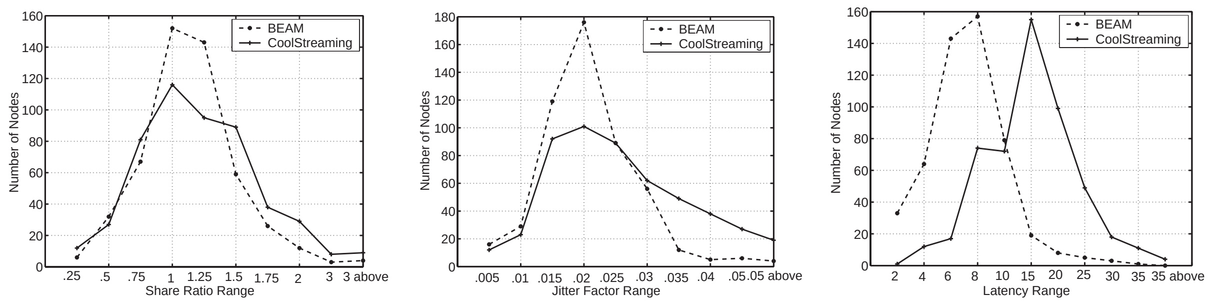
(c) % Uplink Utilization vs Bitrate

Figure 4.2: *Effect of bitrate on the QoS parameters for BEAM and CoolStreaming*

uplink bandwidth is a must for a scalable system [SLL06]. From Figure 4.1(c) it is clear that uplink utilization increases with the swarm size in both of the systems. BEAM has approximately 7% higher utilization and this could be due to the fact that nodes with higher uploading capacity can effectively use their outgoing bandwidth in their other alliances, while the same may not be true for CS where a node with high uplink capacity may remain under utilized due to insufficient requests from its neighbors. In random peering (CS), neighboring peers may or may not request for pieces in the packer, while in an alliance (BEAM), members share pieces in every packet among themselves, ensuring that there are request for upload almost all the time, this increases BEAM's uplink utilization. An optimal utilization of 1.0 is near impossible because of node heterogeneity and lack of download requests from the low capacity peers.

4.2.2.2 Streaming Rate

We vary the streaming rate from 64 Kbps to 512 Kbps in a 2048 node swarm and expect a comparative deterioration in QoS with increasing streaming rate as the nodes need to procure more content for the same playback time. For lower streaming rates, QoS is expected to be near optimal as additional packets are fetched much before their playtime and chances of jitter and hence latency become negligible. From Figure 4.2(a), the difference in average jitter rate between BEAM and CS is marginal for lower streaming rates but more prominent for higher streaming rates when the systems are subjected to stress test. In Figure 4.2(b), similar trends can be observed for average latency while analyzing the effect of varying streaming rate on BEAM and CS. For the same playback time, a node needs to obtain higher number of packets that incurs additional time overhead. Figure 4.2(c) shows the variation in uplink utilization of both the systems. They both peak around encoding rate of 256 Kbps. A plausible reason could be that at this encoding rate, the nodes are able to cater all the requests at optimum rate and this in turn increases the throughput. Node topology and peering partners are important in analyzing the utilization of bandwidth. Most commercial websites stream at rates between 225 Kbps and 450 Kbps as of 2007. Receiver should have $downlink \geq streaming\ rate$ and sender should have enough uplink to contribute. In our simulation, the bandwidth classes of lowest strata is 512 Kbps, so we have limited our discussion to streaming rate of 512 Kbps. 512 Kbps can be considered as a decent rate, though in near future streaming of DVD quality media will require additional bandwidth.



(a) Share Ratio Range (b) Jitter Factor Range (c) Average Latency Range

Figure 4.3: *Share Ratio, Jitter Factor and Latency Range vs Number of Nodes in BEAM and CoolStreaming.*

4.2.2.3 Fairness

To the best of our knowledge fairness has been undermined in P2P streaming models. [AH00, BHP06] have addressed fairness issues in Gnutella and BitTorrent like P2P systems. Chu et al. [CCZ04] have proposed a tax based model for fairness in P2P streaming. Uplink bandwidth is a sparse and most important resource in P2P streaming swarm and end users resort to methods like freeriding [AH00], whitewashing [LFS03] etc. in order to save their uplink bandwidth. As a result, many nodes upload much more than what they should while others get a free ride. Recent measurement studies [HLL06, AMZ06] confirm the same. In this paper, we quantify fairness in terms of content served by each node (uplink bandwidth) or equivalently by their share ratios. We compute the share ratio of all the individual nodes and analyze the correlation, if any, in the QoS perceived by the nodes. Also, we study the fairness of BEAM and CS towards distributing the load evenly among users.

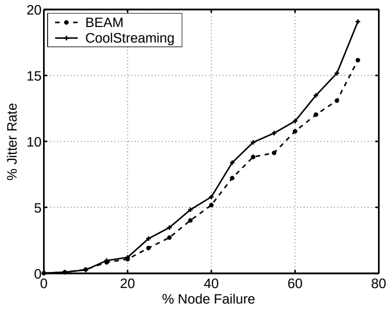
In Figure 4.3(a), we depict the share ratios of nodes and their distribution in BEAM and CS. An ideal share ratio of 1.0 is not possible in such P2P systems due to the node bandwidth heterogeneity [HLL06, AMZ06]. In such cases, the range of share ratios from 0.75 to 1.25 becomes more significant since it is closest to 1.0. Larger the number of nodes having share ratio close to 1.0, fairer is the system. In BEAM, around 1180 nodes out of 2048 have their share ratios in the range 0.75 to 1.25, which forms 57.61% of the total nodes, while the distribution is more spread out in CS with 41.21% nodes lying in the region of share ratios between 0.75 to 1.25.

BEAM encourages user participation, wherein nodes in an alliance exchange pieces, i.e. nodes upload the content to their alliance members while completing their individual downloads. For example, in an alliance of 5 nodes, if one of the nodes procures a packet (either from the server or through some of its other alliances), it forwards the content among its other four members of the alliance. These four members must exchange their pieces amongst themselves. Therefore, a node downloads 4 pieces and at least uploads 3 pieces in its current alliance which makes its share ratio = 3/4 = 0.75. Further, after downloading the complete packet, it forwards (4 or fewer pieces) the content to its other alliances depending upon the number of requests it has and depending upon if those members have procured some pieces from their other alliances. This explains why BEAM has better uplink utilization and more nodes have share ratio around 1.0. However, some disparity can be seen in Figure 4.3(a) where some nodes upload more than 3 copies while others share less than one fourth of the entire content. This is because there are very few requests made to the low capacity peers , and power nodes distribute multiple copies of the content in the swarm. In case of CS, there are more nodes with higher share ratios and comparatively lesser nodes with share ratio closer to 1. This could be attributed to the fact that some nodes with high bandwidth always remain forwarding nodes, i.e. they upload much more than the lower bandwidth nodes either because of excess bandwidth or the topology of the node in which flow could be top-down.

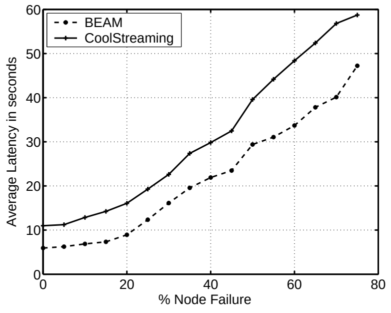
As shown in the Figure 4.3(b), the average jitter factor is 0.0158% and average latency is 15.12 seconds in BEAM, whereas they are 0.0221% and 22.11 seconds respectively for CS. It can be seen that most nodes in the swarm receive average values of QoS parameters for both the systems, i.e. the nature of graphs in Figures 4.3(b) and 4.3(c) are similar and show that nodes contributing fairly to the swarm receive the streaming content with an average jitter factor and latency. This can be seen in Figure 4.3(b) and Figure 4.3(c) where certain nodes have larger latencies and comparatively higher jitter rates. These nodes have lower contribution in the swarm. In BEAM, more than 80% nodes have jitter factor in the range of 0.001 to 0.003 and similarly more than 70% nodes have latencies in the range of 10 to 20 seconds. In case of CS, jitter factors are more spread out as compared to BEAM and there are many nodes with a high jitter factor of 0.040 and above. The initial bootstrapping time in CS is also higher as shown in Figure 4.3(c).

4.2.2.4 Robustness and Reliability

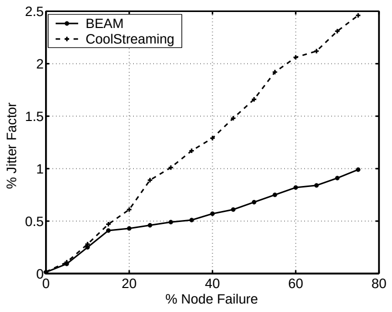
We conducted two types of experiment to evaluate the robustness and reliability of both systems: 1) We injected various percent of node failures after 50% of the simulation run time. 2) We injected one third of node failures at three different intervals: 25%, 50% and 75% of the simulation run. After the complete simulation run, we recorded the QoS factors viz percentage jitter rate and



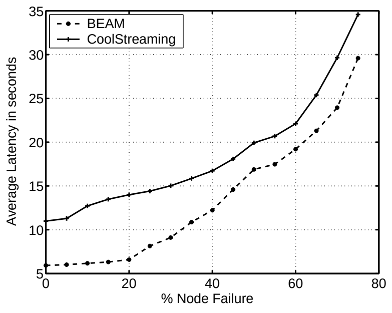
(a) % Jitter Rate vs % Sudden Failure



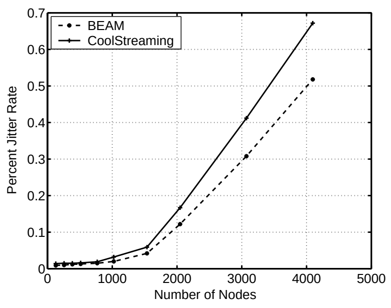
(b) Average Latency vs % Sudden Failure



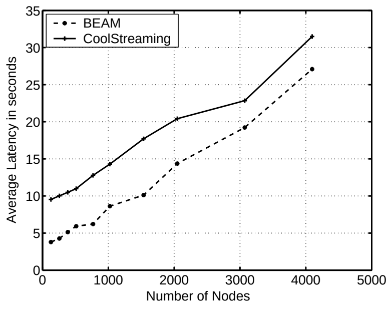
(c) % Jitter Rate vs % Gradual Failure



(d) Average Latency vs % Gradual Failure



(e) % Jitter Rate for Larger Swarm



(f) Average Latency for Larger Swarm

Figure 4.4: Various Workloads of Robustness and Scalability in BEAM and CoolStreaming.

average latency. We study the impact of node failures on these metrics and the overall system performance. This simulation run comprises of 2048 nodes and maintains a 512 Kbps streaming rate.

Figure 4.4(a) shows the results for our first set of experiment where we inject node failures after 50% simulation run. It can be seen that for 0% node failure, the jitter rate is almost negligible for both the schemes. The jitter rate steadily increases to 1% for around 20% node failures. After injecting 50% node failures, we observe that the rise in jitter rate is steep and reaches 8% for BEAM and 10% for CS. This effect can be understood considering the impact of node failures on the alliances. As the number of nodes gradually decline in case of node failure or departure, number of alliance members (h) decrease, thereby weakening the overall graph connectivity. As nodes continue to falter, the alliance becomes sparse and sometimes is broken completely . The increase in time and consequently jitter rate is due to the time required to find alternate paths and receive the content. CS has displayed similar trends except that it has comparatively more jitters with failure. This can be similarly understood that a node requires more time to find new peer with the available pieces.

Figure 4.4(b) shows the impact of node failures on average latency. In BEAM, for 25% node failures, the average latency incurred is around 30 seconds. At 0% failure, the latency is well within 20 seconds and steadily increases with the node failure rate. The steep curve is prominent after 20% failures. Similar is true for CS except that it takes more time to start the media playback. The same average latency is maintained for the rest of the session. A plausible reason for the behavior of BEAM is that since a node cannot procure packet, it issues multiple requests to the node having desired content in the alliance. In case of a complete alliance failure, the nodes need to re-form an alliance, thereby increasing the average latency. In CS, for 25% node failures the latency increases to 56 seconds and rises steadily after that to almost 170 seconds for 75% node failure. Finding new peers after a node failure incurs an additional time in CS and this delay becomes the end to end latency. The difference in results of BEAM and CS in Figures 4.4(a) and 4.4(b) can be understood in the light of SWN, which show robust behavior during churn.

In the second set of experiments we study the effect when node failure is gradual and occurs over three distinct time intervals: 25%, 50% and 75% of the simulation run. This setup depicts more realistic scenario and facilitates easy recovery. For example, to depict 30% node failures, we inject one third i.e. 10% node failures at 25%, 50% and 75% of the simulation times respectively. Figures 4.4(c), 4.4(d) depict the jitter rate and average latency for the above mentioned node failure rates. We observe that for 75% node failures the jitter rate is still under 1% for BEAM, though it has gone considerably high up to 2.5% for CS. The average latency for 75% node failure is around 60 seconds in BEAM and 70 seconds for CS. For lesser percent of node failures, the jitter rate and average latency are found to be under acceptable range of QoS. We observe that in the case where the node departure is gradual (at specific time intervals), the node recovery is comparatively easy and makes system inherently more stable since more time is available for recovery. For example, when the node failure occurs say within the first 25% of the simulation run time, the system is recovered much before another failure occurs at 50% simulation run time. Similarly, when another failure occurs at 75% simulation time the system is already stable.

4.2.2.5 Scalability

In this section, we extend the results obtained for QoS and uplink utilization to larger swarms. We evaluate scalability in terms of maximum number of nodes a streaming system can support without degrading the QoS. In this experiment, we vary the number of nodes from 128 to 16384. From Figure 4.4(e), we observe that for a swarm size of 16384 nodes, the average jitter rate is around 0.0278% in BEAM and almost around 0.04% for CS. With increasing nodes , jitter factor decreases and becomes steady after 1500 node count and marginally increases for very large swarms. However, even with a steep rise in the number of nodes in the swarm the average jitter factor is found to be under acceptable levels. The difference between CS and BEAM is more evident for larger swarms.

From Figure 4.4(f), in BEAM, the average latency is under 30 seconds for a 16384 node swarm. The peer lag is approximately than 20 seconds. As the number of users in the swarm increase, there are more alliances and as the content is forwarded from one alliance to other, the number of total hops increase resulting in a higher latency. As mentioned previously, a high average latency is not acceptable in live media streaming as live media content is time sensitive and loses its importance if the delay is greater. CS and BEAM have a comparable performance except that CS takes a little more time to bootstrap. One of the important problems in CS like models is the high peer lag for media playback and high buffering time. BEAM has displayed considerable improvement in both aspects, i.e. reduced peer lag and reduced initial buffering time from more than half a minute to

Table 4.3: *A comparison of QoS for various $h,\ k$ values for a 1024 node swarm, media encoded with 512 Kbps. Peering denotes peering scheme in BEAM in terms of h and k, N denotes number of neighbors, $BEAM_J$ and CS_J denote Average Jitter Rate for BEAM and CS, $BEAM_L$ and CS_L denote Average Latency for BEAM and CS, $BEAM_U$ and CS_U denote Uplink Utilization for BEAM and CS.*

Peering	N	$BEAM_J$	CS_J	$BEAM_L$	CS_L	$BEAM_U$	CS_U
$h, k = 4, 2$	6	0.0158	0.0221	15.12	22.11	90.13	80.64
$h, k = 5, 2$	8	0.0156	0.0213	15.89	21.89	91.26	82.76
$h, k = 4, 3$	9	0.0162	0.0202	16.23	20.27	92.42	84.34
$h, k = 6, 2$	10	0.0164	0.0206	17.63	22.18	90.57	85.10
$h, k = 4, 4$	12	0.0164	0.0210	16.11	23.34	88.26	84.14
$h, k = 5, 3$	12	0.0159	0.0210	15.04	23.34	92.53	84.14
$h, k = 4, 5$	15	0.0176	0.0231	17.72	23.42	86.47	83.59
$h, k = 6, 3$	15	0.0177	0.0231	17.14	23.42	89.41	83.59
$h, k = 5, 4$	16	0.0186	0.0245	17.98	24.03	85.53	80.68
$h, k = 4, 6$	18	0.0181	0.0244	18.31	24.16	86.77	82.71
$h, k = 5, 5$	20	0.0190	0.0249	18.68	26.98	87.63	84.93

approximately 20 seconds. It is very difficult to achieve TV like switching because of the lack of a dedicated proxy during initial buffering time.

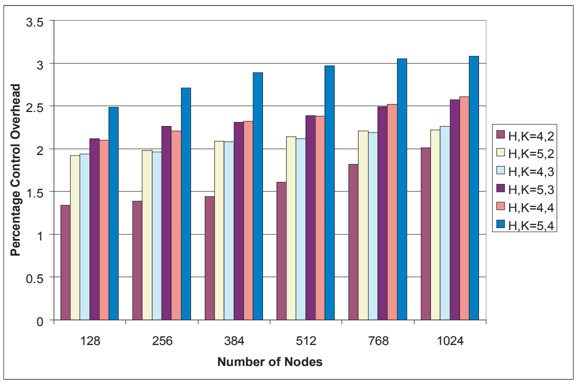


Figure 4.5: *Control Overhead in BEAM for various values of h and k.*

4.2.2.6 Control Overhead

Control overhead is the ratio of the total number of bytes expended in communication and control to the total bytes used for streaming data payload. An efficient system aims to minimize the control overhead (CPU time and bandwidth) and maximize resource utilization towards the streaming content. Figure 4.5 shows the communication overhead incurred in varying swarm sizes ranging from 128 to 1024 nodes. We vary the values of h and k. Recall, that values of h and k denote the node degree. With higher node degree, additional resources are needed resulting in an increased communication and control overhead. For 1024 node swarm and $(h, k) = (5, 4)$ (node degree=16), the control overhead is slightly over 3%. For most other permutations of h and k values, the control overhead is around 2%. Table 4.3 shows affect of various values of h and k on the QoS parameters. We found $(h,\ k) = (5,\ 2)$,$(h,\ k) = (4,\ 3)$ and $(h,\ k) = (5,\ 3)$ as well performing schemes. In our experiments in the paper, we use $h = 4$ and $k = 2$, to show a comparison with CS which has a neighbor count of 6. As mentioned in [ZLL05], for a 200 node swarm in CS, the control overhead is around 2% for node degree of 6, which is quite comparable to BEAM. BEAM and CS almost incur similar overhead. In BEAM, a node sends announce request as it receives a packet, while in CS nodes send information packets to all their neighbors periodically about their buffer state.

Table 4.4: *Evaluation of power nodes and their effect on the QoS factors. The swarm is composed of 2048 nodes and media encoded with 512 Kbps. α is the weight of CSR for calculating the UF. pNode denotes number of distinct power nodes during the streaming session. For various values of α we evaluate number of distinct power nodes active and their effect on overall QoS.*

α	**pNodes**	**Average Jitter**	**Average Latency**	**Uplink Utilization**
0.00	76	0.0175	17.12	90.18
0.20	73	0.0185	17.31	89.46
0.25	67	0.0186	18.23	91.42
0.33	63	0.0175	17.66	89.45
0.50	54	0.0174	17.11	88.29
0.67	48	0.0160	16.54	91.37
0.75	45	0.0158	15.12	90.13
0.80	36	0.0162	15.78	89.27
1.00	29	0.0182	17.95	86.22

4.2.2.7 Effect of Power Nodes

Table 4.4 shows the effect of power nodes on the whole system. For various values of α (the weight factor for CSR and TSR), the jitter factor, average latency and uplink utilization are compared. $pNodes$ is the number of distinct nodes that were chosen as power nodes at least once during the streaming session. The optimal choices were found to be $\alpha = 0.67$ and $\alpha = 0.75$, though other choices were also good with marginal overhead in jitter rate, latency or uplink utilization. This may lead to a very important question: Is it necessary to change the power nodes at all during the streaming session? The answer could depend on many factors. What incentive do the high capacity nodes have in contributing the content altruistically? What if the already chosen power nodes decrease their uploading rate (in the absence of such a policy where best performers in terms of uploading are chosen as power nodes)? We believe that changing the power nodes brings altruism from the high capacity peers who have an interest of being served from the server.

Altruism has a very important effect on the overall efficiency of the swarm and sometimes even more than *tit-for-tat* and any kind of forced fairness policies [PIA07].

4.3 Summary

Simulation results have shown that BEAM performs at par with CoolStreaming and in most cases outperforms it. As discussed in the previous chapter, BEAM's performance improvements mainly stem from its alliance based network topology which is more efficient than a random based network topology. The main improvements are in reducing the average latency, improving effective uplink bandwidth usage and delivering near optimal QoS. It also has shown scalable behavior and is very robust during node failures. In very sparse networks it has shown behavior like any other random network, and the main reason is that because during such times it cannot form any alliances. Formation of small world network gives network stability during node failures and also delivers streaming bytes in near optimal paths. Simulation results in this chapter have corroborated with our hypothesis.

CHAPTER 5

REDUCING CROSS ISP TRAFFIC

We saw in the previous chapter that our proposed model for streaming has shown improvement over the CS model in terms of QoS, uplink utilization, robustness and scalability. In this section, we are going to emphasis on the cross ISP traffic that floods the current day Internet. According to some studies, P2P forms approximately 60% of the Internet traffic as of 2007. Many ISPs bare the brunt of home users that use excessive P2P applications by paying extran revenue. As a result, many ISPs have started traffic shaping. They do deep packet inspection of packets and if there are P2P headers found, the packet is dropped or sometimes its priority is lowered. In USA Comcast ISP does traffic shaping. Rogers is an ISP in Canada, that does the same. As a result, the QoS further degrades at the user end, because in its current most P2P streaming models have random peering mechanism generates excessive cross ISP traffic. In this chapter, we show that using our alliance based peering scheme, we can indeed reduce cross ISP traffic.

5.1 Current Issues

Most P2P streaming algorithms display a greedy behavior in choosing peers and generate excessive amount of cross ISP traffic [AMZ06], thereby increasing the operating cost of an ISP significantly. To overcome such losses, some ISPs impose traffic throttling, where they limit the bandwidths of such P2P traffic. QoS perceived at user end is affected in such scenarios. Cross ISP traffic can be significantly reduced by using a biased neighbor selection policy [BCC06] in BitTorrent like P2P file sharing system and still achieve near optimal performance like BT. In such a systems, a node chooses more peers based on locality of the peers (i.e. within the same ISP) and also chooses some far located peers for content diversity. But BT is a file sharing system, has different internal policies and mainly works on *tit-for-tat* mechanism. We focus on the problem of P2P media streaming cross traffic and propose a preferential peering technique using alliance theory to counter the same and study if reducing cross traffic affects QoS. What are the optimal conditions to achieve both the goals, or it is not possible at all?

5.2 Alliance Based Preferential Peering

As a node joins a swarm, the tracker intelligently assigns it a peerlist using two main criteria: 1) Peers in the similar bandwidth range, if available. 2) Peers in the same ISP, if available and possible. A node while creating and joining alliances can effectively choose its peers or alliance members based on locality and peer bandwidth range. This serves two important purposes: 1)

Cross ISP traffic is reduced. 2) Improving traffic locality ensures less probability of congestion within the interior of the network as compared to bandwidth congestion in the cross ISP links. A good mix of peers from the same ISP and peers in comparable bandwidth range can yield good performance in terms of reducing the cross ISP traffic but it may hurt QoS and other parameters.

5.3 Simulation Results

In this section, we present a set of simulation experiments where there are 20 ISPs and nodes in the swarm belong to these ISPs. We attempt to find and compare percent of traffic between different ISPs and what is internal traffic within the ISPs. We consider two cases: 1) Using BEAM with regular alliance theory.2) Using preferential peering and alliance theory as explained above. The number of nodes vary from 500 to 4000 (i.e. in each ISP, the number of nodes vary from 25 to 200), the streaming rate is 512 Kbps, (h,k)=(4,2). We assume that there is a link between all ISPs and communication can be carried out between all the links.

Fig 5.1 depicts the cross ISP traffic in both the scenarios. Preferential peering using alliance theory reduces the cross ISP traffic significantly. With increasing node count, the percent cross traffic has reduced in both the schemes. While using no peering (i.e. just using regular alliance theory), cross ISP traffic has decreased with increase in node count because the nodes increasingly have better connections within the same ISP, though these connections are not intentionally created in the same ISP. While using preferential, the cross traffic has reduced considerably. For a 4000 nodes swarm, there is a reduction of around 13% in cross ISP traffic, which is approximately 230 GB (a significant volume considering 4000 nodes and a streaming rate of 512 Kbps).

Figure 5.2 depicts the QoS parameters between the two scenarios. It is evident that preferential peering hurts the QoS factors, though, it reduces cross ISP traffic significantly. For smaller swarms the difference is negligible but for larger swarm, which is of interest to us, their difference is conspicuous. For a 4000 node swarm, there is an increase in percent jitters by around 0.2%, and latency has gone up by approximately 8 seconds. Such behavior is understood in terms of nodes not getting the best connection. Since more peers are clustered in similar ISPs, nodes miss out on useful connection which can fetch newer pieces, resulting in extra jitters and latency. Contrary to the intuition that uplink utilization in preferential peering should be better than normal alliance based peering, uplink utilization has indeed gone down in the preferential peering scheme. This is because nodes in the same alliance are not able to get new stream content through the best possible connection. As a result the nodes are idle at various times and the uplink bandwidth is not as effectively used as it is used in normal alliance based peering where a node receives best connection, though at the expense of high cross ISP traffic. It is evident from the figures that achieving good QoS and reducing cross ISP traffic are independent goals and pursuit of one affects other.

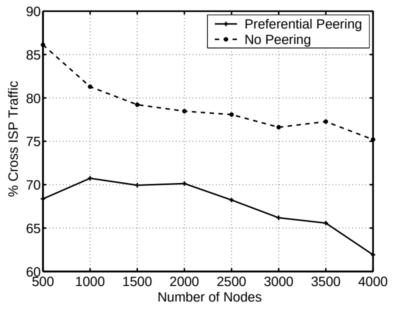
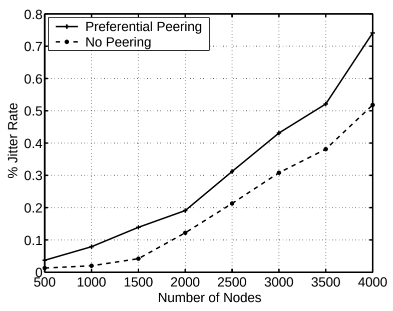
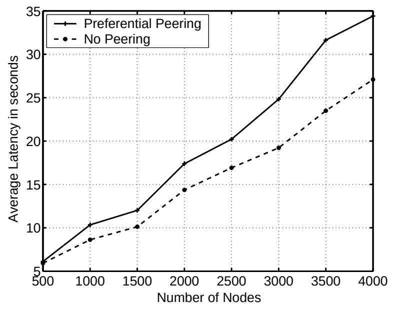


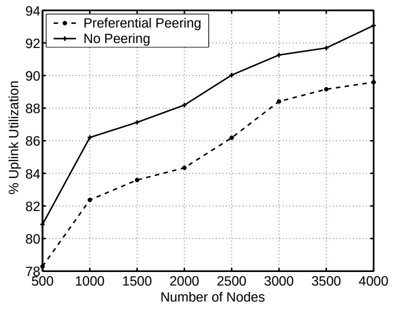
Figure 5.1: *Percent Cross Traffic with Preferential Peering and No Peering.*



(a) % Jitter Rate in Preferential Peering and NO Peering



(b) Average Latency in Preferential Peering and NO Peering



(c) % Uplink Utilization in Preferential Peering and NO Peering

Figure 5.2: Cross ISP Traffic Analysis and its effect on QoS

5.4 Summary

In this chapter, we enumerated the problem of excessive cross ISP traffic in current day internet due to P2P traffic. Current models based on random network topology don't make peering connections in a topology aware fashion and it causes a huge cross ISP traffic problem. ISPs in turn throttle the P2P traffic that further deteriorates the QoS of P2P environments. Using BEAM and preferential peering scheme, we saw that we can considerable reduce cross ISP traffic. It is not possible to reduce much more than that because in that case peers would not have any interesting data to share within their alliance members. Cross ISP traffic brings in interesting data to peers which they can propagate within their alliance. This is an active field of research among ISP to alter their revenue model such that they need not throttle the P2P traffic as it is used for legitimate applications these days like sharing source code of major Linux distributions.

CHAPTER 6

PREFERENTIAL PEERING METHODOLOGY

In this chapter, we analyze the current day challenges in BitTorrent (BT) like P2P systems and propose an alternative model to BT to overcome the problems. Bit Torrent (BT) [BHP06, Coh03] is one of the very popular P2P systems used for bulk file download. Results [BHP06] showed that BT scales very well to large number of users and achieves near optimal performance in terms of uplink utilization, mean download time and fairness. The BT like P2P model is based on tit-for-tat policy, however, it lacks in terms of fairness. Some nodes end up uploading more than 6 copies while others almost get a free ride [AH00]. Our model aims to achieve the following.

1. Survivability: At all times, every block of the file exists in the swarm ensuring system survivability.

2. Minimum Mean Download Time: The model aims to minimize the average of the download times of all the nodes in the system.

3. Link utilization: The model aims to maximize total uplink utilization of nodes in the swarm.

4. Fairness: No node is forced to upload more than it has downloaded.

However, there is always a trade off involved when we try to accomplish these goals. We define stricter bounds to achieve good uplink utilization and minimize the mean download time, which in turn ensures fairness. We propose a preferential and strata based pairing scheme among the nodes. The idea is to group nodes with similar bandwidths together in a single stratum. The tracker issues every node a list of its preferred peers based on their respective stratum. Later on we show that this scheme of assigning the peers indeed is better than random selection of peers as in conventional BT model. We aim to achieve optimal system performance in terms of fairness, uplink utilization and mean download time. The emphasis is overall system benefit and not individual node gain. However, if free riders [AH00] exist in the system, it in turn affects overall system performance. As a result the contributing nodes suffer in terms of fairness. We propose the use of public key cryptography to alleviate this issue and prevent nodes from cheating.

Our main contributions are a preferential and strata based clustering scheme to group nodes and use of public key cryptography to prevent the nodes from cheating. We introduce the notion of TokenFromTracker and Published Upload Speed for the same. We have proposed a self healing and a self punishing model to dissuade selfishness and propagate altruism.

6.1 BitTorrent like P2P Models

BitTorrent (BT) [BHP06, Coh03] is a P2P application used for bulk file download. Conventional P2P systems were used for small sized data files with one to one connections possible between them. Large multimedia files and software distributions demand a fast and efficient mode of transfer. It exploits the uplink speed of end users while they are downloading parts of the file. With BT, files are broken into small chunks typically 256 Kb. As the fragments are distributed to the peers in a random order, they can be reassembled on a requesting machine. Each peer takes advantage of the best connections to the missing pieces while providing an upload connection to the pieces it has already downloaded. This scheme has proven particularly useful in trading large files such as video, games and software source code. In conventional downloading, high demand leads to bottlenecks as demand surges for bandwidth from the host server. With BT, high demand can actually increase throughput as more bandwidth and additional seeds of the completed file become available to the group. Cohen [Coh03] claims that for very popular files, BT supports about a thousand times as many downloads as HTTP and prevents server crashes evident in the HTTP downloads.

There are two types of nodes in the system. The nodes that have finished downloading the file and willingly offer uploads to other users are called as seeds. The nodes still in the process of downloading file are called as leeches. Leeches also offer uploads to other users as they download.

To share a file using BT, a user creates a .torrent file, a small metafile that contains the information like filename, size, hash of each block in the file, the address of a tracker server and miscellaneous data like client instructions. The .torrent file is distributed to the users via some medium like email or website. The original user who is willing to offer the upload starts as a seed while other users start as leeches. Once a new user joins the system he contacts the tracker to obtain a list of 40 peers including seeders and leeches who are in the swarm. A new node upon receipt of peer list contacts these nodes to obtain the file blocks. The nodes in the peerlist which are already in the system send their buffermaps to this new node. Buffermap is a list which contains the list of pieces they currently have. New node then requests the pieces from these nodes. If the peer list goes below 20 due to departure of some nodes the new node makes another request to tracker to give him the addresses of some more peers. It then sends a *have* message to its neighbors about this new piece, so that other nodes can now request for this newly obtained piece.

Every node downloads as many blocks and as fast as they can. For each available source, the node considers the blocks of file available and then requests the rarest block among the peers. This is called as Local Rarest First(LRF) policy. The least replicated block is chosen and downloaded to maximize the content diversity in the system. This makes it more likely that peers will have blocks to exchange. As soon as the client finishes importing a block, it hashes the block to ensure that the hash matches with the hash value in the torrent file. It then looks for someone to upload the block.

A tit-for-tat policy is enforced to make sure that leeches do not get a free ride and also give back to the system by performing uploads. BT gives the best download performance to the nodes with maximum upload, a property known as “leech resistance”. It discourages leeches from downloading the file without uploading it to anyone. This policy forces everybody to contribute to the system to get maximum system benefit. Every node tries to limit the number of uploads at one instant to some small number say 5 to avoid having lots of competing TCP connections [BHP06, Mor97].

A technique called choking is used to limit the number of uploads. A node uses choking to block the upload connections to maintain its own performance. In general, the set of neighbors that a node is uploading to may differ from the set of nodes it is downloading from. Time to time every node performs optimistic unchoking which helps new user to get started. When a new user joins the system there is no way he can start downloading the blocks unless there is optimistic unchoking. The scheme helps the node realize if there is any other node giving better upload speed to him so that it can choke some other connection and unchoke him. Though BT is a good protocol for a broadband user, it is less effective for dial up connections, where disconnections are common. On the other hand, many HTTP servers drop connections over several hours, while many torrents exist long enough to complete a multi-day download.

This process of uploading and downloading continues till nodes procure a complete copy of the file. Once a node finished it can either stay in the swarm and offer its uplink to helps others or it can leave the swarm. The average download time of the users in the swarm is proportional to the number of seeds present and also on the contribution of other nodes.

6.2 Related Work

There has been considerable work done ever since Cohen [Coh03] first came with the idea of BT. Many simulation and analytical based studies have been reported till date. Most simulation based studies focused on BT performance at various setups. Izal et al. [IUB04] focused on the tracker log obtained from the Redhat 9 Linux Distribution. Their work enumerated the basic properties of the torrent i.e. most clients after finishing the download tend to stay in the pool for another 6.5 hours because they need manual intervention to close the BT client and stop uploading. They also reported average upload speed achieved during the run of the torrent. They have seconded the claim by Cohen [Coh03] that tit-for-tat policy is effective in BT and gives good results. Pouwelse et al. [PGE05] also performed a study on a 8 month log obtained from a real life tracker of more than two thousand global components. Their main finding is that within P2P systems a tension exists between availability which is improved when there are no global components, and data integrity, which benefits from centralization. Sherwood et al. [SBB04] have explained the ”Slurpie” system which is very similar to BT. It uses an available bandwidth estimation technique. All nodes downloading the same file contact the topology server. Using the information returned by the topology server the nodes form a mesh and propagates progress updates to other nodes. Slurpie protocol has been implemented and is available for download. Shrivastava et al. [SB05] have presented an incentive based streaming in P2P environment. Qiu et al. [QS04] modelled BT using fluid flow and conducted an analytical performance study. They have derived expressions for average number of seeds, leeches and download time using the node arrival and departure rate. They have shown that BT is scalable and performance improves as there are more users in the system.

Our work mainly focuses on classifying nodes in various strata so that peers mostly exchange packets with peers in the same bandwidth stratum. This is very crucial for good uplink utilization. On the fairness front, we employ a published based model where a node publishes his standard upload speed. To stop cheating, tracker creates a token called “TokenFromTracker” which uses public key cryptography to encrypt the token.

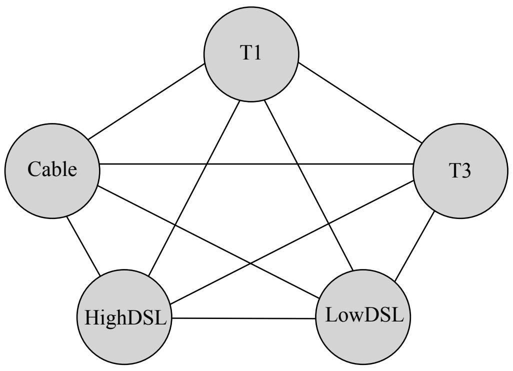


Figure 6.1: *A Strata based mechanism*

6.3 Preferential and Strata Based Model

We propose a novel scheme to classify and group nodes in a similar bandwidth range into respective stratum. Nodes within different strata have the flexibility of communicating with each aother for the requested file block. However, the nodes that lie within the same stratum are the primary preferred peers for file exchange followed by the nodes in the nearby stratum in terms of their bandwidth difference.

In real life scenario we have nodes with heterogeneous bandwidths such as T3, T1, Cable, High DSL and Low DSL. We assume five bandwidth classes related as:

$$\Delta_1 > \Delta_2 > \Delta_3 > \Delta_4 > \Delta_5$$

Let,

$$\Delta_{(i,j)} = |\Delta_i - \Delta_j|$$

where Δ_i and Δ_j are the bandwidths of strata i and j. Therefore,

$$\Delta_{(i,j)} \leq \Delta_{(i,k)} \quad \text{for} \quad |i-j| \leq |i-k|$$

where

$$i \leq j \leq k \text{ or } i \geq j \geq k$$

In our model, a node with bandwidth Δ_2 is most likely to interact with nodes from Δ_2 as $|i-j|$ is 0. A second preference is given to Δ_3 and Δ_1 bandwidth nodes, which are the strata with the second minimum bandwidth difference.

$$\Delta_{(2,2)} < \Delta_{(2,3)}$$
$$\Delta_{(2,2)} < \Delta_{(2,1)}$$

Some fraction of the peer list will include nodes from remote strata. For example, in our case a node with Δ_1 speed would also have Δ_3, Δ_4 and Δ_5 bandwidth nodes as its peers. In our simulation we consider 60% peers from the same stratum, 15% each from the neighboring strata and 10% from remote strata. It can be denoted as $(60, 15, 10)$. We also consider other possible distributions like $(70, 10, 10)$ and $(50, 20, 10)$. Later we show that this scheme of assigning peers is better than random selection of peers (used in conventional BT model). It ensures near optimal uplink utilization as maximum interaction is found to be among nodes within the same strata.

Moreover, this scheme tends to be fair as the nodes form a symbiotic association and are equally benefited in terms of the volume of content served. This forms a natural incentive for the nodes to be in the best possible strata.

Generally such P2P systems comprise of large number of nodes. Nodes can enter and depart at any point of time. Moreover, the node behavior cannot be trusted. There is always a possibility of free riders in the swarm who selfishly download the content without contributing fairly to the swarm. We emphasize on the overall system performance and not on individual gains. Consequently the contributing nodes suffer on account of these free riders. This also has significant impact on the uplink utilization and fairness ensured by the preferential and strata based scheme. We propose a self healing and self punishing model to counter this issue.

We introduce the notion of Published Upload Speed (PUS) for a node. This is to dissuade the node from cheating and to ensure that the node offers the same upload speed as published throughout the time it is in the swarm. Every node upon arrival contacts the tracker. It sends PUS it is going to offer to the peers. An end user (node) can configure the BT client based on his preferences. Even if a node has good uplink speed he would not want to dedicate all his uplink speed. However, if the node publishes a lower uplink speed he will be placed in the lower stratum. So the node resorts to cheating by initially offering high PUS and later configuring his BT client to downgrade the uplink speed. Open source programs for BitTorrent gives an end user a chance to modify the protocol in the code. New programs like Azureus [Azu] can be customized as per user needs and allow user to choose upload speed, number of connections etc. This facilitates cheating by an end user. If a node wants to downgrade his PUS for some reasons it is expected that the tracker is informed and the node obtains a new TokenFromTracker with the new downgraded PUS. This helps the tracker differentiate the cheating nodes from the non cheating ones.

We propose to create a token called “TokenFromTracker”. This token is encrypted with tracker’s private key. The payload contains Node ID, PUS and Arrival Time:

$$\{\textbf{[Node ID, Published Upload Speed, Time]}\mathbf{K_R}\}\mathbf{K_U}$$

For example, node A and B agree to communicate. They exchange their tokens and get the information of peer’s PUS and other details by decrypting the token with tracker’s public key. While the session is on, the peer can gauge the node’s offered upload speed. A node is immediately caught if it keeps upload bandwidth low for a certain period of time interval. The model associates some tolerance T with the upload speed i.e. if a node publishes an upload speed of U, then its uplink speed in the range U-T to U are all acceptable. If it falls below U-T the other node would wait for a small time interval t and eventually disconnect. Peer notifies the tracker about the cheating node. Tracker issues a warning to the cheating node. If number of complaints exceed the threshold K, the tracker brands him as a bad node and throws him out of the swarm. This is a consequence of our self punishing policy. Nevertheless, the warned users who cooperate with the protocol are allowed to stay in the swarm. This is in accordance with our self healing policy. Thus, our model is a self healing and self punishing one and turns the node behavior from selfish to altruistic enhancing the overall system performance.

After acquiring the peer list the node contacts the peers for the first file block. Once it procures the first file block it starts uploading as well downloading other file blocks. For the subsequent blocks Local rarest first (LRF) policy is enforced as in the conventional BT protocol [Coh03]. The nodes continue to exchange the file blocks until they finish their respective downloads. Nodes may volunteer to stay in the swarm or leave. Nodes intimate the tracker while leaving the swarm.

6.4 Simulation Setup

We present the details of the simulation setup for our proposed preferential and strata based model. We present an evaluative comparison against the BT protocol. We have mainly focused on average time to finish the download, fairness in terms of the volume of content served and uplink bandwidth utilization. Under the assumption that $(downlinkspeed > uplinkspeed)$, the bottleneck in most cases is the uplink speed. In such cases downlink speed cannot capture the correct notion of bandwidth utilization. To justify our fairness claims we have taken into account share ratio of the nodes in the system. We compute the variance of share ratios and compare with BT to identify how our model works in case of free riders. Can we lower the disparity of share ratios of the nodes so that free riders have no incentive in their behavior? We have performed experiments to evaluate our self healing and self punishing model.

We implemented a discrete event custom simulator in Java. As mentioned in [BHP06] network propagation delay is relevant only in the case of small sized packets such as request packets. Most

P2P traffic is the data payload and ignoring propagation delay does not have a significant impact on the simulation results. For experimental purposes we make an implicit assumption about the network propagation delay and do not model it in our simulation. We do not model the TCP congestion and delays within the network. The bottlenecks are assumed to be either the uplink or downlink bandwidth and not any other point in the network. We believe that bulk file transfers lasts over an extended period of time and ignoring TCP dynamics for small intervals do not affect the simulation results. Pouwelse [PGE05] findings are that the real world torrent downloads do not necessarily follow any particular arrival pattern. The arrival pattern of nodes in the swarm is assumed to be under Poisson distribution which is closest to the real world compared to any other distribution.

In our simulation setup we have varied the following parameters: Number of users (N) from 128 to 8192, File size (S) from 256 MB to 2048 MB. Each file block is considered to be 256 KB. Initial seed is considered to be a powerful node capable of very good upload speed say 6 Mbps. The various bandwidth strata we have considered are (10000Kbps, 5000Kbps), (8000 Kbps, 4000 Kbps), (3000 Kbps,1000Kbps), (1500 Kbps, 384 Kbps) and (784 Kbps, 128 Kbps), where the first member of the tuple is the max download speed and the second is the max upload speed. The distribution of nodes among various bandwidth strata is uniform. On an average, every stratum has around 20% of the total nodes in the swarm. An implicit assumption is that as nodes finish their downloads they leave the swarm. We have injected around 25% of users who stay in the swarm to help others finish their download. The above mentioned numbers have been obtained from real life torrent examples [IUB04].

We evaluated our model for the three main metrics viz. (i) Average download time (ii) Uplink utilization (iii) Fairness in terms of share ratio of each node in the swarm. Mathematically these are denoted as follows.

1. Average download time $= \frac{\sum_{i=0}^{N} D_i}{N}$,

   where D_i is the download time of node i and N is the total number of nodes in the swarm.

2. Uplink utilization $= \frac{\sum_{i=0}^{N} UT_i}{N}$,

   where UT_i is the ratio of the uplink bandwith used to the uplink bandwith avaliable for node i.

3. Share ratio $= \frac{U_i}{D_i}$,

   where U_i and D_i denote the uploaded and downloaded contents for node i.

Further, to evaluate our self healing and punishing model we injected around 10% cheating nodes. These nodes mimic real world cheating nodes that do not adhere to the protocol. Implementation program detects such cheating nodes for their selfish behavior. We analyze the node behavior during the course of the simulation and quantify the number of nodes that turn from selfish to altruistic.

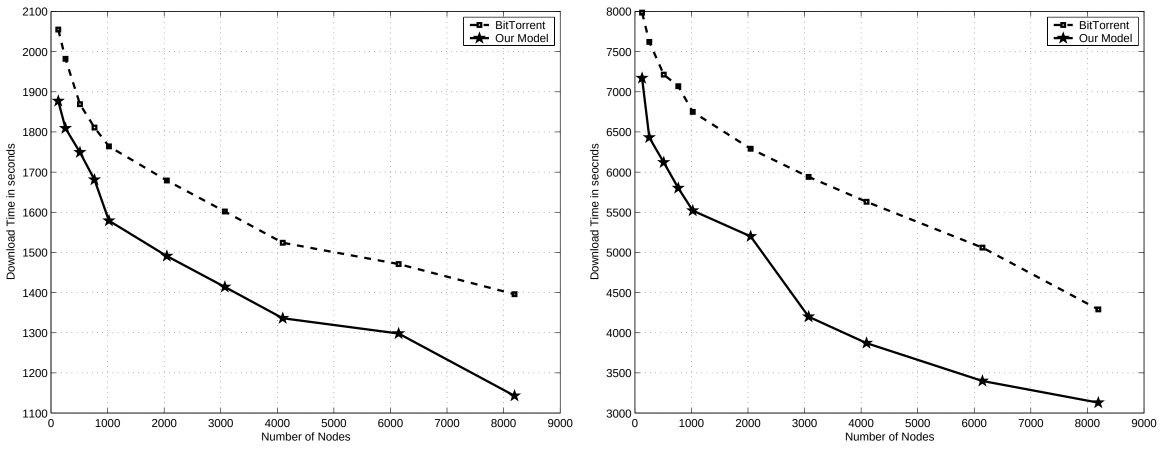


Figure 6.2: *Comparison of Mean Download Time in Our Model versus BT for 256 MB and 1024 MB.*

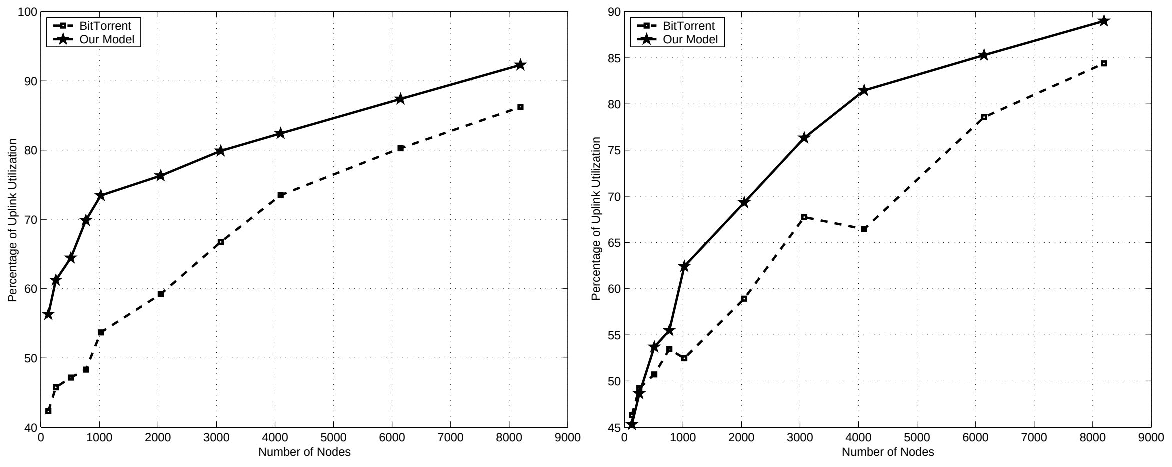


Figure 6.3: *Comparison of Uplink Utilization in Our Model versus BT for 256 MB and 1024 MB.*

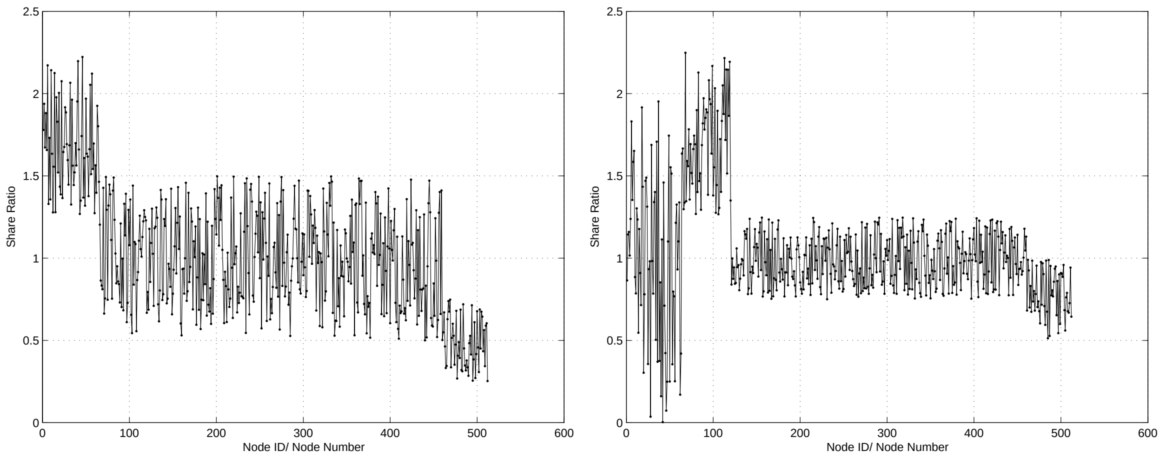


Figure 6.4: *Comparison of Share Ratios in Our Model versus BT for 512users.*

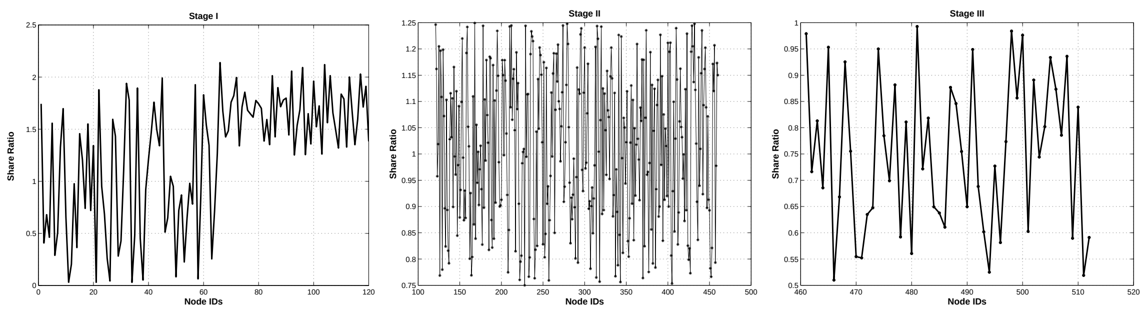


Figure 6.5: *Distribution of Nodes and Share Ratio in Stages I, II and III.*

6.5 Results and Discussion

Our main objectives were to evaluate and compare the mean download time, fairness and percentage uplink utilization. Figures 6.2 and 6.3 show the comparison of Mean Download time and Uplink utilization in our model versus the conventional BT. Figure 6.2 is plotted for file size 256 MB and for 1024 MB. The simulations were run for users ranging from 128 to 8192 and time is calculated in second. It is evident from both the figures that the mean download time in our model has been considerably minimized. Furthermore, with increasing number of nodes, the mean download time decreases in our model. Thus, our model is scalable.

Figure 6.3 represents the percentage utilization of uplink bandwidth for file sizes 256 MB and 1024 MB. For both the cases our model has better uplink utilization compared to the conventional BT. For file size of 256 MB the utilization factors for BT and our model are 86% and 91% respectively while for 1024 MB the utilization factors are 83% and 88% respectively. Our model has consistently scored over BT for all number of users for the both the cases.

We have quantified fairness in terms of Share Ratio. Share Ratio is the ratio of uploaded volume content to the downloaded volume content. Share ratio of 1 is considered healthy and optimal where a node downloads a copy of the file and also gives back to the system the downloaded copy. Typically, original seed ends up uploading number of copies and there are always some very reliable seeds (not original) which stay in the swarm for a while to help others complete their downloads. Such seeds have a very high share ratio typically more than 5. We believe that a node should not be forced to stay in the swarm to finish the download. If nodes stay in the swarm, their altruism is welcome but we believe that user altruism should not be forced. In the figure 6.4 we have depicted the fairness in terms of share ratio. This simulation run was done for 512 nodes in the system. It is evident from figure 6.4 that the variance of share ratio in our model is less as compared to BT. The variance of our model is 0.115725 while BT has 0.156253 which proves that our model is more consistent than BT in terms of fairness. The fact that our model kicks out free riders helps improve fairness among user nodes. The results manifest that our model has achieved the above mentioned goals.

6.6 Analysis

In previous sections we compare our model to conventional BT and conduct an analysis of the same. We begin with quantifying the differences in the tracker overhead in our model and conventional BT. While in the conventional BT the peer list is assigned randomly, in our model the tracker finds best peers for every node. This task is computationally inexpensive as it has to search for peers in the stratum corresponding to the node's published upload speed. The search space is considerably reduced. Tracker monitors the nodes in various strata and uses it to assign the peer list. Tracker creates a token called "TokenFromTracker" and encrypts with its private key. For instance if the Node Id is 64 bits, the published upload seed and miscellaneous information occupy another 64 bits, the tracker encrypts around 16 bytes for a single node. This encryption and computation can be done in parallel while computing the peer list. This does not incur any additional overhead. In addition to this the tracker also logs *bad node history* of the cheating nodes. Tracker warns a node if it gets complaints about him not adhering to the protocol. After monitoring that particular node, if the node behavior persists, the tracker may remove him from the swarm. Eventually the number of nodes the tracker has to monitor decrease. Above mentioned changes are not computationally expensive and do not flood tracker with lot of messages. They can be performed very well without any delay and degradation in the tracker performance.

We perform an analysis of the node behavior during our simulation. There are three stages in the run of the protocol shown in Figure 6.5.

Stage 1: There are not enough number of users in the swarm. Due to less number of users in each stratum, the tracker cannot assign the peer list based on our protocol. Essentially, our model behaves the same as conventional BT in this phase for lack for nodes. This phase does not demonstrate any improvement in terms of uplink utilization as we cannot achieve preferential pairing of nodes. Howsoever, this phase does not dominate the total run time of the protocol and hence does not show a significant impact in the overall results. Figure 6.5 depicts this stage. It is evident that in this stage disparity dominates in the share ratio of nodes.

Stage 2: There are considerable number of users present in the swarm. This state is called "Steady State". This is the phase where our model is most dominant. The tracker allocates peer list based on preferential grouping in accordance with our model. As a result, good uplink utilization is achieved. Since nodes with similar bandwidths are involved in block exchange share ratio is close to 1. This ensures overall fairness in the system. This phase is marked by maximum transitions from selfish to altruistic nodes shown in Figure 6.6.

Stage 3: There is dearth of nodes again because the nodes that have finished the download leave the swarm. The seeds willingly offer uploads to help other users finish their downloads. That is why the share ratio of users involved in this phase is less than 1 and decreasing. It is evident that the share ratio steadily falls below 1.

Second phase dominates 90% of the protocol run time. Any improvement in this phase will be reflected in the overall results. First phase is similar to the conventional BT but its share is very less compared to the second and third phases that dominate the total run of the protocol. Results in the previous sections have demonstrated this fact. These changes are reflected in all three measures namely mean download time, fairness and uplink utilization.

In our simulation run, we injected around 10% cheating nodes. These nodes over a period of time turn altruistic during the simulation run. The nodes that cheat despite the warning are taken off the swarm by the tracker. In Figure 6.6, 900 nodes start as altruistic and 124 as selfish nodes. Towards the end 31 nodes are blacklisted and thrown out of the pool and rest 93 turn good. This shows that our model indeed turns user behavior from selfish to altruistic. In Figure 6.6 the number of altruistic nodes increase while cheating nodes decrease as they are thrown out of the swarm. This shows that our self healing and punishing policy holds good.

6.7 Summary

BitTorrent is inherently a very efficient protocol for bulk file transfer. But it does not achieve the best performance in terms of mean download time, fairness and uplink utilization. We present a

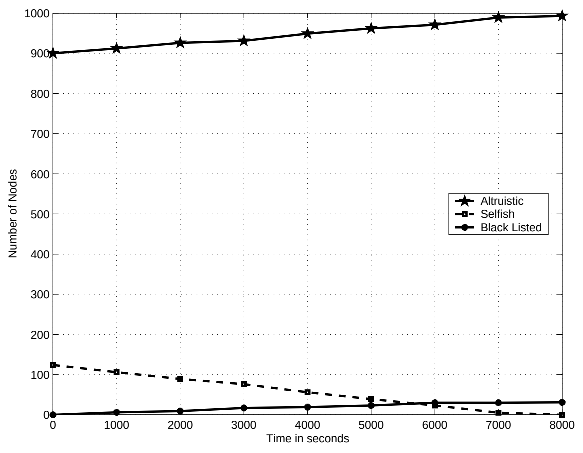


Figure 6.6: *Number of Selfish Users who turned Altruistic.*

refined model by adding strata for various bandwidth users for better pairing between the peers. Results show that this way of assigning peers is better than random selection of peers. Publish based model is an efficient way of classifying nodes in strata. Usage of public key cryptography adds flexibility and is a cost efficient solution to prevent cheating. Our analysis and simulation results have confirmed that our model is stable, scalable and performs well on all the three important metrics. Our self healing and self punishing policy helps turn user behavior from selfish to altruistic. Our results are promising and inspiring. Future goals related to this work are to analyze the graph theoretic properties such as node degree, max flow problem from node to sink of the BT like P2P system.

CHAPTER 7

IMPROVING SECURITY IN BITTORRENT

BitTorrent has shown to be efficient for bulk file transfer, however, it is susceptible to free riding by strategic clients like BitTyrant. Strategic peers configure the client software such that for very less or no contribution, they can obtain good download speeds. Such strategic nodes exploit the altruism in the swarm and consume resources at the expense of other honest nodes and create an unfair swarm. More unfairness is generated in the swarm with the presence of heterogeneous bandwidth nodes. Many high capacity peers contribute much more than needed while low capacity peers contribute very little or nothing. In this research, we propose and investigate new anti-strategic policies that could be used in BitTorrent to minimize the free-riding by strategic clients. In our proposed anti-strategic model, nodes obtain a *token* from Tracker upon joining the swarm which they use while interacting with peers. The token contains information such as *published upload speed*, arrival time and node ID, and this token is signed by tracker such that other nodes can verify the information but nobody can forge it. Other anti-strategic policies include, using a smart tracker that denies the request of strategic clients for peer list multiple times, and black listing the non-behaving nodes that do not follow the protocol policies. These policies help to stop the strategic behavior of peers to a large extent and improves overall system performance. Moreover, in this paper, we also quantify and validate the benefits of using bandwidth peer matching policy. Peers are given a peer list based on their bandwidth range upon arrival. This fosters better uplink utilization, reduces the time for nodes to find the optimal peers for exchanging data and has positive effects on many important metrics like download time, fairness index etc. Our simulations results show that with the above proposed changes, uplink utilization and mean download time improves considerably. It leaves strategic clients with little or no incentive to behave greedily. This reduces free riding and creates fairer swarm with very little computational overhead. Finally, we show that our model is self healing model where user behavior changes from selfish to altruistic in the presence of the aforementioned policies.

7.1 New Security Issues in BitTorrent

BitTorrent (BT) [Coh03] has emerged as one of the most popular peer-to-peer (P2P) models in recent years for bulk file sharing. It shows improved performance in terms of uplink utilization and mean download time as compared to other P2P systems [BHP06]. However, BT is prone to strategic attacks as shown by BitTyrant [PIA07] and others [HP05, LNK06, LMS06]. It also suffers from free riding problem [AH00] and creates unfair swarm. Many high capacity peers[1] upload much more than it is required while many get a free ride. The share ratio (ratio of uploaded volume to downloaded volume content) has shown to vary from 0 to almost 6 in many studies [BHP06].

[1]We have used the terms nodes and peers interchangeably.

The strategic clients exploit the excess bandwidth in the swarm provided by some altruistic peers present in the swarm. They game their BT client in such a way that with very little or no contribution, they can obtain a good download speed. This exploit comes at the expense of other non strategic users. They procure a peer list in excess of 200-300 by constantly querying for more peers from the tracker. Their main strategy is to exploit the optimistic unchoke by seeds, and if there are many seeds, they could benefit without contributing much. While interacting with other nodes in the swarm, these clients gradually decrease their uplink speed while they get service from other peers. If the other peer drops the service because of less uplink speed, such strategic clients increase the uplink speed so as to reach the minimal uplink speed needed to induce cooperation from the other peer. Such clients use many more TCP connections than mentioned in the reference BT client implementation to exploit the maximum download speed for a given uplink speed.

In this paper, we propose a set of new features that, when incorporated, could make BT resistant to such strategic attacks. We determine the impact of these policies on other important factors in the system like mean download time, uplink utilization and fairness towards an end user. There is a trade off involved in accomplishing these goals simultaneously, i.e., minimal download time and fairness do not go hand in hand and pursuit of one affects the other. The BT protocol can be customized in many different ways, where achieving optimal mean download time is one end of the spectrum and achieving fairness the other [BL06]. We investigate as to what point in the whole spectrum of these parameters, could yield a near optimal results with no strategic attacks and high fairness. We also investigate the effect of altruism, i.e., self volunteers who offer their upload speed in return for nothing, on the swarm performance.

The main goals of BitTorrent like P2P system are the following. Each one of them is very important independently and not completely mutually exclusive with others. In our present work, we propose new policies that will make them robust and capable of overcoming the current problems of strategic attacks.

- Survivability: Every block of the file must exists in the swarm at all times so as to ensure that all nodes can finish the download at some point of the time, better sooner. BT employs *local-rarest-first* policy for replication of pieces and has shown to be very efficient [LUM06].
- Download Time: Every node individually attempts to finish its download as soon as possible. Selfish clients use greedy policies for the same. Our aim is to keep the mean of download times of all the nodes to as less as possible.
- Uplink Utilization: High uplink throughput is desirable for scalable system and partially reflects the mean download time. Peers at every point of time attempt to find a partner which has high uploading capacity. Seeders upload the content to the peers with high download speed to effectively improve the uplink throughput, which improves download time. In later and new version of BT, the seeds uniformly distribute the content to the nodes rather than few high capacity nodes.
- Fairness: The swarm should be fair i.e., no node should be forced to upload much more than what it has downloaded. No one should be able to get a free ride. Voluntary altruism is welcome for the swarm.

- Robust: The swarm should be robust to strategic clients and must not let these clients to download selfishly at the expense of other non strategic nodes.

In this chapter, we propose new features for BT to overcome strategic attacks and to improve the overall system performance. In particular our main contributions and findings are:

1. We use anti-strategic policies to guard BT against selfish clients [PIA07, LMS06]. To this end, peers exchange tokens given to them by tracker for keeping tab of uplink speed of the other peer. Intelligent tracker prohibits strategic clients from procuring peer list multiple times. Nodes that upload garbage content are quickly identified and blacklisted by neighboring nodes. These policies are resistant to strategic attacks and does not let the clients degrade performance of other peers.

2. We quantify and experimentally validate the concept of bandwidth peer clustering [PG06, BHP06, LUM06] and show that it shows significant improvement in uplink utilization and mean download time.

3. We show that altruism is indeed very important for improving the overall system performance. Altruistic swarms finish the download much faster and use the resources near-optimally when compared to swarms without altruism.

Section 6.1 gives an overview of BT like P2P model. We present the details of our model in section 7.2. Section 7.3 describes the metrics we use for evaluating our model and the simulation setup. We briefly describe the metrics which we use in our simulations and comparisons.

And finally results are in section 7.4. In section 7.5, we present the related work done for the enhancement of the BT protocol.

7.1.1 Strategic Attacks on BitTorrent

P2P systems are inherently based on user altruism and participation [BHP06] but this concept is exploited by free riders who do not want to contribute their uplink speed and are only interested in downloading. Free riders in BT resort to acts like tuning their client for minimum upload or even strategizing the client for maximal download for minimum contribution. Liogkas et al. [LNK06] proposed 3 main techniques to exploit BT: 1) Download from seeds. 2) Download from peers with high uplink capacity and 3) Advertising false pieces. Locher et al. [LMS06] proposed to procure huge peer list so that a client can afford to only interact with the seeds. BitTyrant [PIA07] is a client implemented using Azureus [Azu], and has been very successful in exploiting the vulnerabilities of BT. BitTyrant exploits the altruism present in the swarm by procuring a large peers list in the swarm. It then attempts to establish connections with seeds and procures free content. While interacting with other nodes, BitTyrant adopts a policy of uploading minimum content to get the maximum download. One of the very important exploits BitTyrant used is to avoid equal split policy while uploading, rather it uploads in different fractions to its peers to get the maximal download. While interacting with a peer, it gradually decreases its uplink speed as long as it gets reciprocation. If the other peer chokes BitTyrant then it increases the uplink speed and determine the level of participation (uplink speed) needed to keep the connection alive and get the downloads.

BitTyrant does this over many connections and attempts to maximize the download for a given uplink speed limit. BitTyrant by virtue of its greedy policy does bring performance improvement for a client, but it can hurt the swarm performance if all the peers use the BitTyrant client [PIA07]. This would mean BitTyrant improves performance for an individual user and not for the whole swarm. In this paper, we investigate if such strategic behavior could be alleviated in the swarm while achieving near-optimal behavior in the overall swarm performance. To this end we propose new set of anti-strategic policies mentioned in the next section.

7.2 Our Proposed Policies

In this section, we propose a set of features for BT that could help to overcome the strategic attacks and improve the swarm performance.

7.2.1 Anti-Strategic Behavior

We propose to use anti-strategic behavior in every BT client in the wake of recent attacks which leads to poor swarm performance [PIA07]. In particular, we propose to use the following three strategies to overcome strategic attacks.

7.2.1.1 Token From Tracker

We introduce the notion of *Published Upload Speed* (PUS) for a node. This is to dissuade the node from cheating and to ensure that the node offers almost same upload speed as published throughout the time it is in the swarm. Every node upon arrival submits to the tracker the PUS it is going to offer to the peers. The tracker creates a token, called *TokenFromTracker* ($[Node\ ID,\ PUS,\ Time]K_R$), which consists the Node ID, PUS, and its arrival time, and encrypts it with its private key (K_R). While interacting with peers in the swarm, nodes exchange their $TokenFromTracker$. By decrypting the other peer's token ($\{[Node\ ID,\ PUS,\ Time]K_R\}K_U$) using the public key of tracker (K_U), they get an estimate of the uplink speed the other node is offering.

Assuming a node splits its uplink bandwidth equally among its local peers (say 5), a node can gauge which peers give better uplink speed and it can unchoke the connections to those peers. While a session is on, nodes can resort to cheating by initially offering uplink speed equal to PUS and later on gradually decreasing the uplink speed (similar to BitTyrant). A node can be immediately caught if it keeps upload bandwidth low for a certain period of time interval. The model associates some tolerance T with the uplink speed i.e. if a node publishes an upload speed of U, then its uplink speed in the range $(U - T)$ to U is acceptable. If it falls below $(U - T)$ the other node would wait for a small time interval t and eventually disconnect. The tolerance T and t takes care of the network anomalies that might arise and a non strategic user should not be punished for it. Peer notifies the tracker about the cheating node. Tracker warns such nodes after h complaints (say $h = 3$) and upon receiving k such complaints (say $k = 5$), tracker bars the node from getting any more peer list and blacklists such node, and intimates the nodes in its peer list about the existence of such cheating node. This is a consequence of the punishing policy of our model. Nevertheless, the warned users who cooperate with the protocol, i.e. after warning stop the strategic behavior are allowed to stay in the swarm. This is in accordance with our self healing policy.

An end user (node) can configure the BT client based on its preferences. A node can have a good uplink speed, but it would not want to dedicate all its uplink speed to BT application. The node can resort to cheating by initially offering high PUS and later configuring its BT client to downgrade the uplink speed. Open source programs for BT give an end user a chance to modify the protocol in the code. New programs like Azureus [Azu] can be customized as per user needs and allow user to choose upload speed, number of connections etc. This facilitates cheating by an end user. If a node wants to downgrade its PUS for some reasons, it is expected that the tracker is informed and the node obtains a new $TokenFromTracker$ with the new downgraded PUS. This helps the tracker differentiate the cheating nodes from the non cheating ones. Nodes intimate the tracker while leaving the swarm.

7.2.1.2 Smart Tracker

Most strategic attacks stem from the fact that such clients request for more peers from the tracker. After every small time interval, they request for additional peers till they have peers in excess of 200-300. They then launch strategic attacks by interacting with more peers and also hope to get optimistic unchoke from high capacity peers. We propose to use a *smart tracker*, wherein for every request for additional peer list by a client, tracker checks if it has 40 live peers in the swarm. If no, it provides it additional peer list so that the client has at least 40 peers in the list, else it rejects its request of additional peer list. This can substantially lower down such strategic moves by clients who are trying to maximize the profit. In the results section, we quantify the effectiveness of smart tracker in alleviating the behavior of such clients.

7.2.1.3 Blacklisting Nodes

To counter the false publishing attack of nodes wherein the client falsely sends 'have' messages of rare blocks in the swarm and in turn uploads garbage content in order to obtain some useful content, we propose to use a policy wherein a node blacklists such a node upon finding that it uploaded garbage content. Future interactions with such nodes are avoided and tracker is made aware of such garbage content. Tracker warns such nodes after h complaints (say $h = 3$) and upon receiving k such complaints (say $k = 5$), bars the node from getting any more peer list and blacklists such node. The selfish nodes in presence of such policy cannot publish false message for long as they will be blacklisted by most nodes in its neighborhood and it will severely hurt its chances of obtaining new and useful content. In the results section, we show that by adopting this policy, significant swarm bandwidth, that goes in downloading such garbage content, can be saved. It also serves as a warning to such nodes that publishes false content and upload bad content not to resort to such techniques. Results show that user behavior indeed changes from selfish to normal upon receiving warning from tracker.

7.3 Evaluation of Our Proposed Model

We present the details of the simulation setup for our proposed anti-strategic and strata based model. We present an evaluative comparison against the BT protocol. The main metrics for comparison are average time to finish the download, uplink bandwidth utilization, and fairness index in terms of the share ratio. We now detail the metrics used for our comparison.

1. Average download time: It is the mean of download times of all the nodes in the system. Mathematically,

$$Average\ Download\ Time = \frac{\sum_{i=0}^{N} D_i}{N}$$

   where D_i is the download time of node i and N is the total number of nodes in the swarm. The download time of high capacity nodes has shown to be less compared to its weaker counterparts [BHP06]. By evaluating the mean download time for the whole swarm, we can get a fair bit of idea about the performance of the BT protocol in the swarm.

2. Uplink utilization: Uplink bandwidth is the most sparse resource in the system. In most realistic scenarios $Uplink\ Bandwidth \leq Downlink\ Bandwidth$ holds good, so we limit

our discussion to only uplink bandwidth. A good uplink throughput i.e. (ratio of uplink used to uplink available) would mean a lot of resource (uplink bandwidth) are pooled in the swarm that can serve peers, which in turn helps to lower the download time. Moreover, some ISPs charge their end users for the bandwidth used per unit time. Such users would want to maximize the uplink throughput for saving the ISP fee. Small et al. [SLL06] proved that maximization of uplink speed leads to scalable systems. Mathematically,

$$Uplinl\ Utilization = \frac{\sum_{i=0}^{N} UT_i}{N}$$

where UT_i is the ratio of the uplink bandwidth used to the uplink bandwidth available for node i.

3. Fairness: We define fairness in terms of share ratio of content served by nodes. Share ratio of end users over the period of complete download depicts the contribution of the nodes quite fairly. In an ideal system, nodes have share ratios of 1.0, where an end user downloads the content and passes on equally to other end users. But given the dynamics of the internet, churn, and peering schemes, it is very difficult to achieve the same in BT. So, more the number of nodes close to share ratio of 1.0, the fairer is the system, i.e., the lesser the variance of share ratios from the mean, fairer is the system. We use Jain's fairness index [JCH84] to

evaluate the swarm fairness.

$$f(x_1, x_2, x_3, \ldots, x_n) = \frac{(\sum_{i=0}^{n} x_i)^2}{n \sum_{i=0}^{n} {x_i}^2}$$

where $x_1,\ x_2, ...,\ x_n$ are the share ratios of the nodes. The value of fairness index varies from 0 (worst) to 1 (best).

4. Altruism: We define altruism as the excess uplink bandwidth provided to the swarm by a node, i.e., a node when willingly uploads more content than what it downloaded is altruism for the swarm. Most altruistic behavior is displayed by the original seed. Altruism can be either voluntary or sometimes circumstantial/forced. Voluntary altruism is welcome as it helps the system with more resources and the load is divided. In forced altruism, a node has to upload more content to even download a single copy of the file (torrent data). Forced altruism is not fair as a node is compelled to upload more than what it had downloaded. We define Altruism as the excess content uploaded to the swarm after reaching a share ratio of 1.0. For e.g., if a node has uploaded 6 copies and downloaded 1, its Altruism factor will be 5.0 since it uploaded 5 excess copies of the torrent data in the swarm. A node with share ratio of 1.0 will have altruism factor as 0. Mathematically,

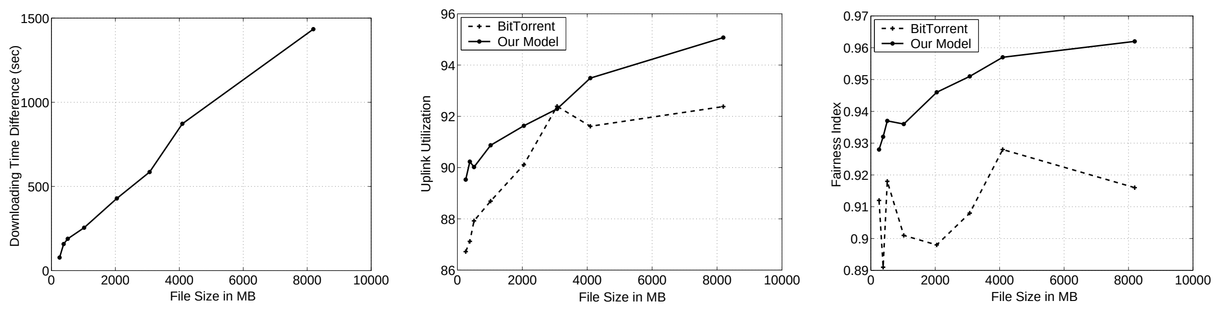
$$\begin{aligned} Altruism\ Factor &= \frac{Uplaods - Downloads}{Downloads} \\ &= Share\ Ratio - 1 \end{aligned}$$

7.3.1 Simulation Setup

Most real world physical links have $(downlinkspeed > uplinkspeed)$, so the bottleneck in most cases is the uplink speed. In such cases downlink speed cannot capture the correct notion of bandwidth utilization. To justify our fairness claims, we have taken into account share ratio of the nodes in the system. We compute the fairness index of share ratios and compare with BT to identify as to how our model works in case of free riders. Can we lower the disparity of share ratios of the nodes so that free riders have no incentive in their behavior? We have performed experiments to evaluate our self healing and self punishing model. We used a simulation based approach, primarily because it is extremely difficult to gauge the behavior of large swarms without the participation of thousands of node. Moreover, simulated settings give flexibility to play with different parameters without affecting the overall behavior.

We used the BRITE universal topology generator [MLM01] in the Top-Down Hierarchical mode to model the physical network topology of Autonomous Systems (AS) and the routers. All AS are assumed to be in the Transit-Stub manner. Overlay is assumed to be undirected. Unlike other simulators [BHP06, MSR05], we assume that the bottleneck in the network can appear in the access links of source and destination (i.e. first-mile and last-mile hops) as well as the non access links that are in the interior of the network, in particular within or between carrier ISP networks. The nodes in the swarm are assumed to be of heterogeneous bandwidth classes namely:(512Kb, 128Kb), (768Kb, 256Kb), (1024Kb, 512Kb), (1536Kb, 768Kb), (2048Kb, 1024Kb) where first and second member of the tuple are the maximum downlink and uplink speed of a node respectively. The distribution of these bandwidth classes is uniform in the swarm. To simulate the congestion in the Internet, we induce 5% congestion in the non access links within the interior of the network. In such congestion scenarios, the available bandwidth to nodes is the minimum of the bottleneck at source or destination and the bottleneck in the non access links. The delay on inter-transit domains and intra-transit domains are assumed to be 100 ms and 50 ms respectively, while delay on stub-transit is assumed to be 30 ms and intra-stub transit links are randomly chosen between 5ms and 25ms. We simulate the TCP level dynamics like timeouts, slow start, fast recovery and fast retransmission by introducing a delay of 10 RTTs [CK01]. We model a flash crowd scenario for the arrival of users in the swarm, i.e. all users are present in the swarm when the file sharing begins, as this is the most relevant and challenging scenario.

In our simulation setup we have varied the following parameters: Number of users (N) from 128 to 8192, File size (S) from 256 MB to 8192 MB. Each file block is considered to be 256 KB. Initial seed is considered to be a powerful node capable of very good upload speed say (6 Mbps). A default implicit assumption is that as nodes finish their downloads they leave the swarm. For some other experiments, we assume that nodes stay in the swarm. We mention the settings at the appropriate places in the text. Further, to evaluate our self healing and punishing model,



(a) Download Time Difference vs File Size in MB (b) Uplink Utilization vs File Size in MB (c) Fairness Index vs File Size in MB

Figure 7.1: *Effect of Anti-Strategic Behavior in Our Model*

we injected around 10% cheating nodes. These nodes mimic real world cheating nodes that do not adhere to the protocol. Implementation program detects such cheating nodes for their selfish behavior. We analyze the node behavior during the course of the simulation and quantify the number of nodes that turn from selfish to altruistic.

7.4 Results and Discussion

We use the above simulator to test the efficacy of our proposed policies. We compare the behavior of conventional BT protocol with our proposed changes. Our proposed changes are to overcome the current day vulnerability in BT which the strategic clients exploit. We test each of our proposed techniques one by one, and then later combine all the proposed changes to see the overall behavior.

7.4.1 Effect of Anti-Strategic Behavior

Current day exploits by strategic clients are: 1) Requesting peer list from tracker after every few minutes. 2) While interacting with other nodes, decrease the contribution gradually to arrive at an equilibrium that will give client maximum download speed for a given upload contribution. 3) Sending false 'have' messages and uploading garbage content. In the first set of experiments, we use the anti-strategic policy to see how well a swarm can cope up with the strategic clients. The main components of our anti-strategic policy are mainly using *TokenFromTracker* that guards honest nodes from such selfish clients that degrade the upload quality, stopping selfish clients from obtaining peer list every few minutes by using a smart tracker and finally blacklisting the nodes that uploads garbage content.

In this set of experiments, we investigate the difference in average downloading time between the conventional BT with our proposed policies. Figure 7.1(a) depicts the results where the torrent file size varies from 256 MB to 8192 MB for a swarm of 1024 nodes. 10% of the nodes behave strategically all the time during the simulation in both the cases. For smaller file sizes the difference is small but with increasing file sizes the difference in download time is very prominent. For 8192 MB file, the difference in average downloading time is as high as 1450 seconds (approximately 40 minutes). Since every node keeps a tab on each of its connection, and as selfish clients degrade the uplink bandwidth, the other node disconnects leaving the selfish client with no option but to increase its contribution. We found that by using *TokenFromTracker*, the BitTorrent *Tit-For-Tat* mechanism is used effectively and the equilibrium of uplinks bandwidths between the nodes is reached earlier as compared to standard BT implementation. Legout et al. [LUM06] and Piatek et al. [PIA07] have showed that BT takes unusually long time to reach the steady state or equilibrium, where peers have found their optimal partners with respect to the uplink bandwidths. This means that non strategic clients do not suffer on account of strategic clients while using *Token From Tracker* as all legit nodes can interact with nodes which are in similar bandwidth range. In the download of a large file like 8192 MB with a large swarm, the difference in download time is evident as the policy of our model are enforced for a longer time. Similarly, the smart tracker denies the recurring requests of strategic clients for more peer list. In this case such strategic client have to make connections within the given peer list and they cannot exploit the best connections through seeds and high capacity leeches. Finally, the strategic nodes that send false 'have' messages for rare blocks for obtaining some useful content are quickly identified by fellow peers by checking the hash of the blocks. Such nodes are warned and denied any help from tracker in obtaining new peer list in future. Over a longer period of time, such nodes are found to adhere to the protocol. All the above factors show a considerable performance improvement in the average download time.

Similarly, the difference in uplink utilization can be understood from figure 7.1(b). Average download time of the swarm is directly proportional to the uplink utilization. By strictly imposing the anti-strategic policy on the strategic clients, we can leverage the uplink bandwidth of such selfish nodes. We see a consistent superior performance of our model over BT by using the anti-strategic policy. Our model on an average has $3-4\%$ more utilization than the BT. This difference amounts to large chunk of bandwidth (approximately 21 MB/s) for a 1024 node swarm and an average uplink bandwidth of 537.6 KB/s (average of all the uplink bandwidths of all the strata in the swarm). The more the uplink bandwidth is pooled in the swarm, the better will be the uplink throughput and hence better will be the average download time.

Figure 7.1(c) shows the fairness index of both the models. The FI has been calculated on the share ratios of the nodes in the swarm. If all the nodes have share ratio 1.0, then the system would be ideal and FI would be 1.0. But as the share ratio deviates from 1.0 the FI goes down. It gives a decent measure of the performance of nodes in terms of contributing to the swarm. Strategic clients consume lot of bandwidth of the swarm with no or little contribution, compelling honest nodes to contribute more. This disparity of share ratio is quite evident in the figure 7.1(c). By using anti-strategic policy, we can eliminate the greedy behavior of selfish clients. The share ratios of the nodes are more even in case of our model. It is extremely difficult to achieve a perfect FI of 1.0, because of node heterogeneity, network topology, churn and altruism. We discuss the effect of altruism in section 7.4.4 as it has a major impact on the results.

7.4.2 Effect of Blacklisting

We quantify the amount of uplink bandwidth that can be saved by simply using blacklisting policy as mentioned in section 7.2.1.3. Figure 7.2 depicts an estimate of bandwidth that is saved in a 1024 node swarm with 10% strategic clients. We assume that such clients on an average upload 5% garbage data. For large torrent file size, the difference in bandwidth is huge, and it adversely affects uplink utilization and hence mean download time of the swarm. While in the case of anti-

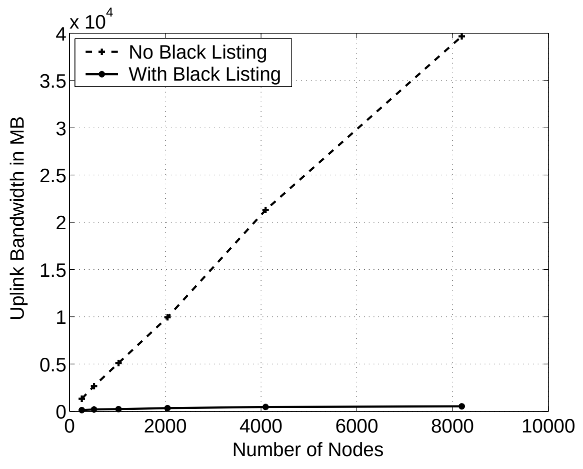


Figure 7.2: A comparison of bandwidth saved using blacklisting policy

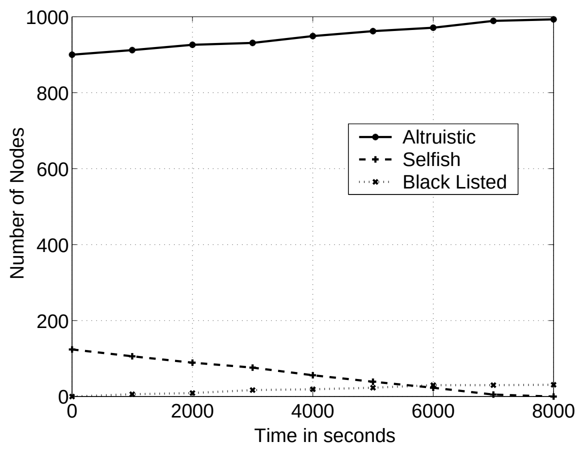


Figure 7.3: An estimate of nodes that turned from selfish to altruistic upon warned

strategic policy, part of the bandwidth is lost until all the nodes that upload garbage are caught and blacklisted. Tracker informs peers of such cheating node not to interact with them. This indeed saves a huge chunk of the bandwidth.

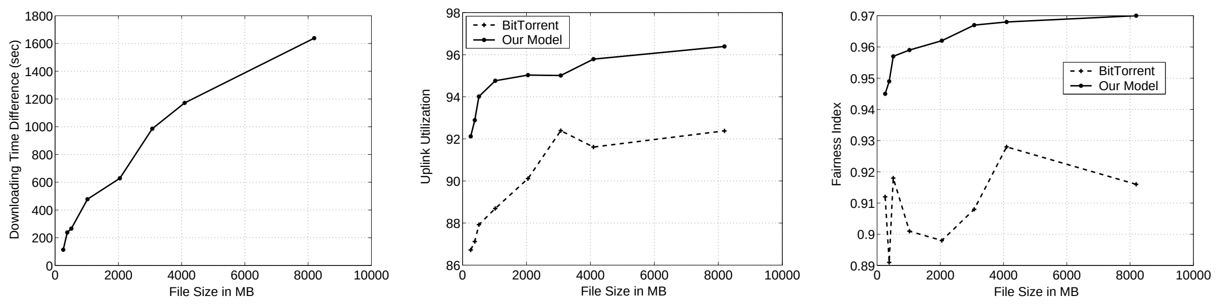
In the next set of experiments, we started out an experiment with 1024 nodes out of which 900 are honest and 124 are strategic. During the simulation run, these nodes behave strategically. Upon being caught and warned by tracker, such nodes either adhere to the protocol or continue cheating.

If they adhere to the basics of the protocol, they are included as honest nodes but upon constant cheating they can be blacklisted and thrown out by the tracker. Figure 7.3 depicts the results. Towards the end of the simulation run, 91 our of 124 strategic nodes turned honest and 31 were blacklisted. By giving strategic clients a chance to improve their behavior, we can leverage their contribution for the rest of the download. Some clients that consistently cheat are thrown because of their leech behavior, they do not contribute any resources to the swarm, which is against the ethics of the P2P file sharing etiquettes.

7.4.3 Overall Effect of Our Model

We showed the effect of each of our techniques individually in sections 7.4.1 and 7.4.2. Now, we apply of these strategies together and compare the result with the reference BT protocol. We assume 10% strategic nodes in 1024 node swarm for various torrent file sizes. We use $(70, 30)$ peering policy for sending peer list to the new incoming node. Figure 7.4 depicts the results after incorporating all our techniques.

The effect of anti-strategic policy and bandwidth clustering adds up to show a considerable difference in the download time difference between BT and our model. As mentioned earlier, uplink throughput and download time both improve as a result of both anti-strategic behavior and bandwidth clustering. A consistent difference of around 4 to 5% in uplink utilization and difference of around 45 minutes in the mean download time is evident from figure 7.4(a) and 7.4(b). Both



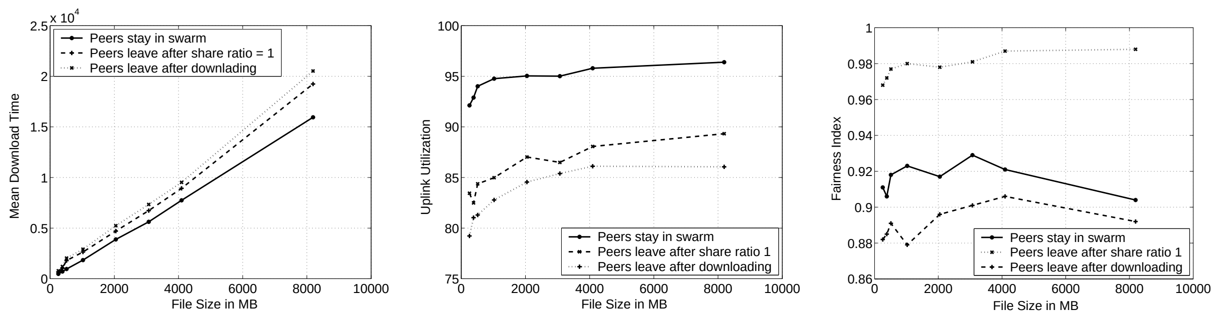
(a) Download Time Difference vs File Size in MB (b) Uplink Utilization vs File Size in MB (c) Fairness Index vs File Size in MB

Figure 7.4: *Overall effect of our Model*

the policies, i.e. anti-strategic behavior and bandwidth clustering do not interfere with each other when used together. Bandwidth clustering is only used for issuing a peer list, while anti-strategic behavior mainly *TokenFromTracker* is built on top of the bandwidth stratum a node belongs to. Policies like Smart tracker and blacklisting are independent of bandwidth clustering and pose no interference to other policies of our model. Similarly, the FI is significantly superior in our model as compared to BT as shown in 7.4(c), signifying that in our model more nodes have share ratios around 1.0 and is comparatively fairer. Though, it is very difficult to achieve a perfect FI of 1.0, we present in section 7.4.4 that in some cases of non altruistic behavior, FI very close to 1.0 can be obtained but at the expense of mean download time and uplink throughput.

7.4.4 Effect of Altruism on the System

As mentioned earlier, one of our main finding is that altruism plays an important factor in improving the download time. There are many seeds who voluntarily offer their upload bandwidth



(a) Download Time Difference vs File Size in MB (b) Uplink Utilization vs File Size in MB (c) Fairness Index vs File Size in MB

Figure 7.5: *Altruism effect on BitTorrent*

in return for nothing, which in fact reduces many peer's download time. To quantify the effect of altruism, we use the altruism factor as described in section 7.4.4. We conducted three experiments: 1) Users are altruistic, i.e. they remain and offer their uplink bandwidth even after they have finished downloading. The $Altruism\ Factor$ of these nodes is strictly greater than 0. 2) Users are not altruistic but they offer uplink till they reach a share ratio of 1.0, i.e. the $Altruism\ Factor$ of these nodes is 0. 3) Users are greedy and leave the swarm as soon as they finish the downloading. The $Altruismfactor$ of such nodes is strictly less than 0. We investigate the trends obtained in the main metrics when we vary our system to these variants.

From figure 7.5(a), we can see the effect of altruism in minimizing the mean download time. High capacity peers when engaged in voluntary uploading the content enhances the swarm mean download time. For a large file of 8192 MB, with 1024 nodes in the swarm, the difference in mean download time between the two schemes wherein peers stay in the swarm and where peers depart immediately after download is close to one hour. This result is quite intuitive as seeds offer many more copies to the swarm taking the load off the leeches, and they can utilize their uplink in obtaining other useful content from other peers, thereby minimizing their download time. The content uploaded by the seed is very important in this context. On the other hand if nodes leave the swarm as soon as they reach a share ratio of 1.0 or immediately after their download, it will be up to the remaining nodes to pool in the uplink resources for all the nodes, thereby increasing the download time. These results reaffirm the notion that altruism is a very important factor in BitTorrent and one of the most important reasons for its enhanced performance. Similar results can be understood for uplink utilization. When seeds stay longer in the swarm and offer their uplink to fully utilize their outgoing bandwidth, the leeches use their uplink bandwidth with other leeches, thereby optimizing the uplink throughput. Same is not true when peers depart after downloading as the remaining peers have to search more for useful content and sometimes remain idle for lack of useful content. This reduces the uplink throughput.

Altruism, although, improves uplink throughput and mean download time, it is not fair to the contributing nodes. The FI is poor in the case when peers stay in the swarm after finishing their own download as the share ratio of seeds exceeds far above 1.0 and many leeches do not even have to upload a copy back to the swarm, i.e. their share ratio is much less than 1.0, creating this unfairness. In the case when peer stay till they upload a copy back to the swarm, most nodes depart the swarm with share ratio 1.0, thereby making FI close to 1.0. In the case when peers depart the swarm soon after download, the disparity is even higher. Only the few altruistic seeds offer uploads to most leeches in the swarm and this increases the disparity and reduces the FI. As nodes leave the swarm, the nodes in the swarm have to search more for the useful content and sometimes remain idle for the lack of interesting data. This reduces the uplink throughput and increases mean download time, though has a very high FI.

It is extremely difficult to reach a standard set of arguments where the behavior of the system is optimal in all the main metrics. Therefore, we try to reach a common ground where mean download time and uplink utilization is near optimal and FI is approximately in the range 0.9-0.95 or better. We show that this indeed can be achieved using the proposed policies in the paper.

7.5 Related Work

In this section, we enumerate previous work done in literature related to security threats in BitTorrent. In section 6.2, a general history of work done in BitTorrent is presented.

Bharambe et al. [BHP06] created a discrete event simulator to test BT on various different parameters. They showed the presence of significant altruism and unfairness in the swarm. They proposed to use TFT at the block level rather than rate based TFT to overcome unfairness. Fan et al. [BL06] in their analytical study showed that BT could be designed in several different ways, where achieving fairness among end users could be one end of the spectrum and minimizing the mean download time the other end. Legout et al. [LUM06] showed that BT's piece replication using rarest first algorithm is efficient and to replace such a policy is not justified in the context of P2P file replication in the internet. They also showed that the newly incorporated choke algorithms in BT induces reciprocation and is robust to free riders. They also showed that choke algorithms is fair and better than bit level TFT.

Shneidman et al. [SPM04] showed that BT indeed can be exploited using Sybil attacks [Dou02] and by uploading garbage content. Other vulnerabilities and strategic attacks on BT have been mentioned in section 7.1.1.

We present the first and foremost work to defend BT against strategic attacks, not previously demonstrated. We perform the study of BT using the proposed anti-strategic policies and come to the conclusion that we can indeed have a fairer and more efficient swarms in terms of optimal mean download time and uplink utilization. The other part of our work validates the improved performance of BT while using bandwidth peer matching policy. We validated this concept using different set of experiments under different settings. Finally, we second the claim of Piatek et al. [PIA07] that altruism is very important for improving overall swarm performance.

7.6 Summary

We presented some defense mechanisms and policies against the strategic BitTorrent clients. In particular, we showed that by using our proposed anti-strategic policies and bandwidth clustering, not only can the system be prevented from such cheating and strategic attacks but also overall system performance in terms of mean download time, uplink utilization and fairness can be improved. Our simulation results corroborate with the proposed theory. Clustering peers of similar bandwidth has shown to be very effective in utilizing the uplink capacity, and it reduces the mean download time. Anti-Strategic policies do not let cheating clients to stay longer in the swarm. They either are kicked off the swarm or they turn altruistic (from selfish), and the uplink resources of such nodes is utilized and is extremely important for the swarm. We believe our results can provide research insights for the development of new defence mechanism in present day BitTorrent clients to guard against the strategic attacks. Moreover, bandwidth clustering of similar nodes can be easily incorporated into the clients straight away as it can be done with very minor protocol changes.

CHAPTER 8

CONCLUSION AND FUTURE WORK

We introduced a novel framework, *BEAM*, that uses alliance theory for peering to solve some of the existing problems in chunk based P2P media streaming. BEAM is also shown to be robust and scalable framework for P2P media streaming and it overcomes the current prevalent problems. In particular, our main contributions and findings are:

1. Peer lag (while media playback) can be significantly reduced from the order of minutes to approximately 10-20 seconds in the swarm. Initial buffering time has been reduced from around 30 seconds to 10-12 seconds for smaller swarms and less than 20 seconds for larger swarms (4096 nodes) in BEAM. Further reduction in such buffering time to a few seconds is extremely difficult because: a) Buffering time requires the time to find the path, stream content and possible forwarders of the stream content. b) Lack of any dedicated proxy during the initial (buffering) period. c) Heterogeneity of node bandwidths in the swarm.

2. BEAM has displayed robust and scalable behavior while delivering near optimal levels of QoS under varying workloads and conditions. Uplink utilization has improved considerably over CS and throughput is more then 90% for larger swarms. Alliance based peering theory

encourages every node to contribute in order to to receive the content, and indeed generates a fairer swarm.

3. Our preferential and anti-strategic strategy improves current day BitTorrent performance. Not only does it improve dowload time, uplink utilization, fairness but also secure network from strategic clients. We also recommed smart trackers to be implemented in BitTorrent for securing BitTorrent swarms.

8.1 Future Work

Our results are inspiring and provides research insights towards development of newer and efficient peering strategies in P2P media streaming systems. In continuation of our work, we enumerate the possible future work in the dissertation.

1. Applying current alliance based peering model to static P2P file sharing and BitTorrent models and evaluate its performance. We have been successfully able to show our alliance based scheme in streaming domain. Static file sharing is trivial compared to streaming, but we foresee that we might need to tweak the alliance based model for it to suit the file sharing mode.

2. Analysing alliance based peering for security related issues in P2P media streaming. With increasing security threats, it is imperative to investigate more in security domain. Some common threats are free riding, whitewashing, nodes sending garbled and malicious payload,

and denial and distributed denial of service (DoS) attacks. We need to analyse and investigate how to port the current alliance based model to counter the above mentioned security threats.

3. Evaluation of our proposed P2P models with dedicated CDN servers which are P2P friendly. It has been shown that P2P models always incur an initial latency, no matter how well the network is setup. We have shown that forming alliance does incur an initial overhead. For data streams to reach all the nodes, find all the paths does incur some overhead in terms of initial latency. If we can have some dedicated proxy like a CDN, the QoS can be much better during initial time period. Later, P2P networks can support the user with data streams. It will be very interesting to see the performance of a P2P network in which CDNs can support the newly arrived nodes with service initially on.

4. A possible future work would be to study such P2P systems from payment based perspective, wherein an under provisioned user pays certain revenue to CDN to make up for weak uplink bandwidth. Such proposal for revenue models are floating around. Most current ISP model work on flat fee for home users in most countries as of 2008. This is not true for larger organization, where they pay revenue based on usage of service. By having a more consistent model for revenue payment, we can enforce fairness in such P2P models. Most current P2P models try to enforce fairness through an *a la carte* model, where an end user gets service in proportion to the uplink offered. But most of these models cannot capture the true dynamics because it is very difficult to provide differentiated service in such random swarm based environments. In such scenarios, revenue based system can help in the deployment of such P2P services in conjunction with CDNs.

5. Using P2P dataset and crawlers to study congestion within the internet. P2P crawlers can serve as a tool for solving other problems in tcp/ip networking. Large datasets are generated while crawling for P2P networks. Such logs can be used to infer many other properties pertaining to congestion within the internet.

LIST OF REFERENCES

[AH00] E. Adar and B. Huberman. "Free Riding on Gnutella." Technical report, Xerox PARC, August 2000.

[AMZ06] S. Ali, A. Mathur, and H. Zhang. "Measurement of Commercial Peer-To-Peer Live Video Streaming." In *ICST Workshop on Recent Advances in Peer-to-Peer Streaming*, Weaterloo, Canadda, 2006.

[Azu] "Azureus." `http://azureus.sourceforge.net/`.

[BBK02] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. "Scalable application layer multicast." In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 32, pp. 205–217, New York, NY, USA, October 2002. ACM Press.

[BCC06] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection." In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, p. 66, Washington, DC, USA, 2006. IEEE Computer Society.

[BHP06] A. Bharambe, C. Herley, and V. Padmanabhan. "Analyzing and Improving a BitTorrent Network's Performance Mechanisms." In *IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[BL06] D. Chiui B. Fan and J. Lui. "The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design." In *14th IEEE International Conference on Network Protocols*, 2006.

[Bol01] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.

[CC05] D. Harrison C. Dana, D. Li and C. Chuah. "BASS: BitTorrent Assisted Streaming System for Video-on-Demand." In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2005.

[CCZ04] Y. Chu, J. Chuang, and H. Zhang. "A case for taxation in peer-to-peer streaming broadcast." In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pp. 205–212. ACM Press, 2004.

[CDK03] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. "SplitStream: high-bandwidth multicast in cooperative environments." In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 298–313, New York, NY, USA, 2003. ACM Press.

[CK01] S. Choi and C. Kim. "Loss recovery time of Impatient variant of TCP NewReno." volume 37, pp. 1559–1560, 2001.

[Coh03] B. Cohen. "Incentives Build Robustness in BitTorrent." In *P2P Economics Workshop*, 2003.

[CRS02] Y. Chu, S. Rao, S. Seshan, and H. Zhang. "A case for end system multicast." *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast*, **20**(8), 2002.

[DBG] H. Deshpande, M. Bawa, and H. Garcia-Molina. "Streaming Live Media over a Peer-to-Peer Network." Technical report.

[Dou02] J. Douceur. "The Sybil Attack." In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pp. 251–260, London, UK, 2002. Springer-Verlag.

[Fei05] "FeiDian." `http://tv.net9.org/indexhtml`, 2005.

[GR05] C. Gkantsidis and P. R. Rodriguez. "Network coding for large scale content distribution." In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pp. 2235–2245 vol. 4, 2005.

[HHB03] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. "PROMISE: peer-to-peer media streaming using CollectCast." In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pp. 45–54, New York, NY, USA, 2003. ACM Press.

[HLL06] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. "Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System." In *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[HP05] D. Hales and S. Patarin. "How to cheat BitTorrent and why nobody does." Technical report, Department of Computer Science, University of Bologna, May 2005.

[IUB04] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, Al A. Hamra, and L. Garcés-Erice. "Dissecting BitTorrent: Five Months in a Torrent's Lifetime." volume 3015 / 2004, pp. 1–11. Springer Berlin / Heidelberg, April 2004.

[JCH84] R. Jain, D. Chiu, and W. Hawe. "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems." DEC Research Report TR-301, Digital Equipment Corporation, Maynard, MA, USA, September 1984.

[Kle00] J. Kleinberg. "The Small-World Phenomenon: An Algorithmic Perspective." In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[LFS03] K. Lai, M. Feldman, I. Stoica, and J. Chuang. "Incentives for cooperation in peer-to-peer networks." In *Workshop on Economics of Peer-toPeer Systems*, 2003.

[Li04] J. Li. "PeerStreaming: A Practical Receiver-Driven Peer-to-Peer Media Streaming System, MSR-TR-2004-101." Technical report, Microsoft Research, September 2004.

[LMS06] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. "Free Riding in BitTorrent is Cheap." In *5th Workshop on Hot Topics in Networks (HotNets), Irvine, California, USA*, November 2006.

[LNK06] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. "Exploiting BitTorrent For Fun (But Not Profit)." In *5th International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA, February 2006.

[LUM06] A. Legout, G. Urvoy-Keller, and P. Michiardi. "Rarest first and choke algorithms are enough." In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pp. 203–216, New York, NY, USA, 2006. ACM Press.

[MLM01] A. Medina, A. Lakhina, I. Matta, and J. Byers. "BRITE: an approach to universal topology generation." In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, pp. 346–353, 2001.

[Mor97] R. Morris. "TCP behavior with many flows." In *ICNP '97: Proceedings of the 1997 International Conference on Network Protocols (ICNP '97)*, p. 205, Washington, DC, USA, 1997. IEEE Computer Society.

[MSR05] N. Magharei, D. Stutzbach, and R. Rejaie. "Peer-to-Peer Receiver-driven Mesh-based Streaming." August 2005.

[net] "NetworkX." `http://networkx.lanl.gov/`.

[PG06] D. Purandare and R. Guha. "Preferential and Strata based P2P Model: Selfishness to Altruism and Fairness." In *ICPADS (1)*, pp. 561–570, 2006.

[PG07a] D. Purandare and R. Guha. "An Alliance Based Peering Scheme for P2P Live Media Streaming." *IEEE Transaction on Multimedia*, **9**(8), 2007.

[PG07b] D. Purandare and R. Guha. "An alliance based peering scheme for peer-to-peer live media streaming." In *P2P-TV '07: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pp. 340–345, New York, NY, USA, 2007. ACM.

[PGE05] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. "The Bittorrent P2P File-sharing System: Measurements and Analysis." In *4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.

[PIA07] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. "Do incentives build robustness in BitTorrent?" In *NSDI'07*, April 2007.

[Pla] "Planet Lab." `http://planet-lab.org`.

[Ppl05] "PPLive." `http://www.pplive.com`, 2005.

[PS02] V. Padmanabhan and K. Sripanidkulchai. "The case for cooperative networking." In *Peer-to-Peer Systems: First International Workshop, IPTPS 2002*, pp. 178–190, Cambridge, MA, USA, 2002.

[QS04] D. Qiu and R. Srikant. "Modeling and performance analysis of BitTorrent-like peer-to-peer networks." In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 367–378, New York, NY, USA, 2004. ACM Press.

[SB05] V. Shrivastava and S. Banerjee. "Natural selection in peer-to-peer streaming: from the cathedral to the bazaar." In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pp. 93–98, New York, NY, USA, 2005. ACM Press.

[SBB04] R. Sherwood, R. Braud, and B. Bhattacharjee. "Slurpie: A Cooperative Bulk Data Transfer Protocol." In *IEEE INFOCOM, March 2004*, 2004.

[SLL06] T. Small, B. Liang, and B. Li. "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming." In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pp. 539–548, New York, NY, USA, 2006. ACM Press.

[SM05] P. Rodriguez S. Annapureddy, C. Gkantsidis and L. Massoulie. "Providing Video-on-Demand using Peer-to-Peer Networks, MSR-TR-2005-147." Technical report, Microsoft Research, 2005.

[sop04] "SOPCast." `http://www.sopcast.org`, 2004.

[SPM04] J. Shneidman, D. Parkes, and L. Massoulie. "Faithfulness in internet algorithms." In *Proc. SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04).*, Portland, OR, USA, September 2004. ACM SIGCOMM.

[THD03] D. Tran, K. Hua, and T. Do. "Zigzag: An efficient peer-to-peer scheme for media streaming." In *Proc. of IEEE INFOCOM*, 2003.

[VIF06] A. Vlavianos, M. Iliofotou, and M. Faloutsos. “BiToS: Enhancing BitTorrent for Supporting Streaming Applications.” In *IEEE Infocom 2006 Global Internet Workshop*, April 2006.

[WS98] D. Watts and S. Strogatz. “Collective dynamics of ’small-world’ networks.” *Nature*, **393**(6684):440–442, June 1998.

[ZLL05] X. Zhang, J. Liu, B. Li, and Y. Yum. “CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming.” In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pp. 2102–2111 vol. 3, 2005.