ECDP: Efficient Content Distribution Protocol

Debessay Fesehaye
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2302
Email: dkassa2@illinois.edu

Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2302
Email: klara@illinois.edu

***Abstract*—In this paper we present ECDP, an efficient incentive based prioritized content distribution protocol. In ECDP content which originates from some servers or peers is first distributed to some users (peers). As soon as the content reaches some of these peers, other peers can get the content either from the servers or any of the peers whichever gives the lowest per packet price and highest throughput.**

An ECDP content information manager (CIM) calculates the weighted (prioritized) fair rate for each stream at each node. To do this the CIM first obtains the upload and download base rates from each peer's rate monitor (PA). A peer selects or looks up a content in the CIM database using a web interface. The CIM selects a content source node (CDN server or peer) which gives the lowest (bandwidth) price, highest rate (throughput) and lowest delay for each requesting peer.

In ECDP, each content source periodically adjusts its bandwidth prices and prioritized rates based on the current demand. Unlike existing incentive-based mechanisms, ECDP enforces the rate allocations by setting the flows' sending rates (congestion window, receive window and maximum window sizes) to ensure efficient and accurate incentive to the participating nodes.

We have implemented ECDP in the NS2 simulator. Simulation results show that ECDP can outperform existing schemes in terms of file download time and throughput. The results also demonstrate that ECDP obtains fair uplink prices for the uploaders and fair cost for the downloaders maintaining an overall system fairness to benefit all peers, content providers and network operators. The results also demonstrate that ECDP efficiently enforces the prioritized allocations. ECDP can be deployed in the current Internet without the need of changes to the TCP/IP stack or routers. We have also implemented ECDP using an Apache SQL Server with PHP in Linux virtual machines and demonstrated that ECDP is a scalable protocol.

I. Introduction

With the fast growth of the Internet and networking technologies, there has been an explosive growth of online content [1], [2]. These online contents are generated either by centralized content providers (Comcast, Amazon, etc) or distributed users (Youtube, Facebook, etc). Such content generation is expected to grow even more (40-45% a year) [2] with the further expansion and sophistication of the Internet and networking technologies. The content can be none real-time ordinary static file (OSC), a realtime 2-dimensional (2D) streaming video content like YouTube video or a 3 dimensional (3D) streaming content [3].

Traditionally, centralized content providers (CCP) use content distribution networks (CDN) to distribute their contents to their customers. CDNs may use their own network as the case of Comcast and Google [2] or they can use other operator networks to distribute the content. Distributed user generated contents (UGC) such as YouTube and Facebook contents are stored in central Google and Facebook servers and then distributed using the respective CDNs. In the UGC case, once users upload their contents to Youtube, Facebook and other similar services, they lose full control of their contents. On the other hand, such companies (Google and Facebook) generate revenues from paid advertisements using the UGC making it inconvenient for users to share contents. Besides, as users view content from such centralized company servers, their privacies are compromised when personal information is shared with third-party websites. These trends of privacy issues and lack of content ownership demand new architectures for content sharing.

As discussed in [4] it is feasible for users to own their contents and share it with others directly from their home. There are newly emerging distributed social network technologies such as the Diaspora (social network) [5], [6] to achieve the goal content ownership and privacy. The Diaspora social network consists a network of nodes or pods. Each node is hosted by many different individuals and institutions. Users (peers) can host their data with a traditional web host, a cloud-based host, an ISP, or a friend. Users (peers) create accounts on any server and can exchange data with users in other servers. Using such distributed networks of peers both UGC and contents by CCP can be shared.

Using cooperative distributed peers has long been used to distribute content over the Internet. In the year 2007 about 30-40% of of Internet traffic was peer-to-peer (P2P) traffic with BitTorrent claiming most of it [1]. As discussed in [1] P2P traffic is growing within access networks after a global decline in the last two years. Besides, the volume of video content is increasing. With the increase in high bandwidth content demands, content providers should either over-provision their bandwidth to handle peak demands or rely on purchased service such as Akamai. However as discussed in [8] it is cheaper for content providers to purchase bandwidth from their users than using third party content distribution networks (CDNs) or purchasing the infrastructure to directly serve contents. Besides assisting CDNs, using P2P networks results in significant scalability gains as discussed in [9], [10].

While using cooperative customer peers to distribute content, providers need to be mindful about incentives to pay back

peers for their upload bandwidth. Besides, content providers need to make sure that the incentives and returns are accurate enough to offer better quality of service (QoS) guarantees. Using an efficient, fair and accurate peer incentive mechanism can also benefit *content providers* and *network operators* significantly. Content providers can save on bandwidth cost by buying peer link bandwidth. Besides, peers who get significant credit (financial or content credit) from uploading content are most likely to subscribe to more contents potentially increasing the content demand. More content demand can also translate into more link bandwidth demand which can benefit network operators.

BitTorrent [11] and its variants [12], [13] prevent free riding by using a Tit-for-Tat (TFT) incentive mechanism. However in these systems two users are incentivized to exchange only if each has content the other wants as also explained in [14]. Price-Assisted Content Exchange (PACE) [15], [14] and Dandelion [8], [16] are other price-based-content exchange schemes. In Dandelion the content provider rewards uploaders with a fixed credit value for uploading a chunk of a content. Downloaders spend a fixed amount of credit units for each chunk they download. Dandelion doesn't consider a dynamic bandwidth price as a function of content demand and resource (uplink bandwidth) supply. On the other hand PACE obtains per uplink prices based on total demand for each file (content) and supply (uplink capacity). However, in PACE demand may be overestimated, as a buy client can issue several sequential requests to different sell clients until a successful download [15]. Hence, PACE does not guarantee fair-exchange of content for payment. Besides in PACE only the price is considered to choose an uploader (server) of a content regardless of whether or not there are other uploaders which can offer a higher rate (throughput). Both PACE and Dandelion do not provide a mechanism where end-to-end rates are allocated and enforced to all flows of the systems efficiently. They also do not consider prioritized rate allocation where some content requests have a minimum rate requirement.

In this paper we present an Efficient Content Distribution Protocol (ECDP) based on accurate, fair and efficient peer incentives. ECDP helps distributed peers exchange content which can be OSC, 2D or 3D data. ECDP consists of 3 main logical components namely *content index manager* (CIM), *prioritized max/min rate allocation* (PRA) and *bandwidth and content pricing* (BCP). These components interact with each other. The CIM consists of databases with information of peers and data contents. The PRA component is done with the help of the CIM and distributed peer agents (PA). It is where prioritized rate is calculated for each upload and download link of the peers and other main content servers. The rates are then used to choose a content source and to set the sending rates of the corresponding flows. BCP which is also done by the CIM and PA is a component where the bandwidth and content prices are calculated adaptively to ensure incentives between the participating peers. Peers which upload more, earn more credit which can be of monetary value or in terms of download bandwidth or content discounting.

In ECDP each participating peer has an incentive to actively maintain and upload contents to other peers. The best price and transfer strategy for each peer is moderated by software components called content information managers (CIM) along with peer agents (PA). In ECDP each peer which wants to participate in the ECDP *automatic fair trading* installs an associated PA. Each PA then sends its peer's settings to its CIM. The peer settings include the up and downlink capacities of the peer, policy preferences, and ID of the content of interest. The CIM first makes authentication and registration of the requesting peer. The content source chooses between *monetary incentive* and *bandwidth incentive* modes while registering. The *monetary incentive* mode allows peers to earn monetary credit for their upload capacities used and pay in monetary amounts for the download bandwidth of other content source they download content from. In the *bandwidth incentive* mode the monetary credit of peers is converted into higher upload bandwidth values from the content sources. A monetary amount to pay of such peers is also converted into the corresponding upload bandwidth reduction from their content sources. In the *monetary incentive* mode, the registered peer maintains paid accounts with the content information manager (CIM) which communicates with the content providers. In this mode, the price we use in ECDP is directly mapped to real monetary value. This real currency approach eliminates the credit depletion issues [15]. For paid contents, the price of the content is added to the bandwidth prices. The CIM then makes initial calculation of the up and down link rates and prices of each peer based on the current demands (requests) and updates its *content index database* (CID). After this, the CIM selects the best content source (uploader) for the requesting peer, based on the content source selection policy which the requesting peer chooses. The CIM informs both the selected content source (uploading peer) and the receiving (downloading) peer of the match along with the uplink and downlink rates.

The source of the content then uploads the content to the destination at the rate which is the minimum of the uplink and downlink rates. The source peer sets its congestion window (maximum congestion window) to the product of its round trip time (RTT) and the minimum of the uplink and downlink rates it obtained from its CIM. Or simply the uploader sets its congestion window or maximum congestion window to the product of its upload rate and RTT. The downloader sets its receive window to the product of its downlink rate and its RTT.

The PA of the uploader also updates its total amount to earn from the upload every control interval. The PA of the downloader periodically checks the amount it has to pay along with the rate at which it is downloading a content. If there is any discrepancy in the download rate either because of some unexpected bottleneck link in the core network or because the uploader is cheating, the PA on the content receiving end sends new rate values to its CIM. The CIM then informs the uploading PA of the new effective rate. The price calculation algorithm is simple and difficult to cheat. In ECDP both

uploaders and downloaders have no incentives to cheat. The uploader losses customers and hence money (or download bandwidth) if it cheats and the downloader losses a cheap and high throughput source if it cheats as each uploader also has the right to disagree (and drop a customer) by informing the CIM. If a source claims to have a higher rate than it has, its bandwidth price decreases accordingly and congests itself. The congestion can also result in flows taking longer to finish keeping the misbehaving source busy at a low bandwidth for longer than necessary. If flows do not finish quickly, the uploader does not get its credit quickly. Moreover, the CIM monitors all the values making it difficult for either content source or destination to misbehave.

ECDP can also be a network operator centric approach where operators and Internet Service Providers (ISPs) can use it to efficiently distribute content and avoid congestion. As every requester is checking and paying for the content it requests, ECDP inherently mitigates Denial of Service Attacks (DoS). In the operator-centric ECDP scheme, there is a CIM associated with each bottleneck link forming a hierarchy of CIMs. At the lowest level of the hierarchy, there are the PAs and then the second level CIMs, and continues to the highest level CIMs. The operator CIM can then choose local uploaders or global uploaders based on its policies.

Unlike previous work [15],[8], the design of ECDP can allocate rates for different content requests using priorities based on their minimum rate requirements. ECDP can also limit the lifetime of the content to a user-defined parameter. These features are specially important for 2D and 3D live streaming contents which have real time requirements. For instance, to render a 3D video, streams should be synchronized and rendered within a short time gap between them.

We have implemented ECDP in the NS2 [17] simulator and using an Apache SQL server with PHP in Linux virtual machines. The NS2 simulator is so robust that descriptions of the streams of the 3D content can be taken as inputs to produce an emulated 3D video as output. The simulation results show how ECDP can outperform existing content distribution schemes in terms of download time and throughput. The numerical results also demonstrate that the different components of ECDP work according to the design. The SQL implementation of ECDP using PHP shows that ECDP can scale to millions of peers and contents.

The main contributions of this work are as follows.

- We have designed an efficient content distribution protocol (ECDP) with efficient cross-layer content source selection and congestion control mechanisms.
- We have shown that ECDP provides accurate and efficient incentive mechanisms to benefit content providers, content users and network operators. The incentive is in real monetary values (*monetary incentive* mode) and can also be translated into download rate (*bandwidth incentive* mode).
- We have implemented ECDP in the NS2 simulator, evaluated the performance of ECDP, showing that it can outperform existing schemes.
- We have experimented with ECDP using Apache SQL server, and have shown how ECDP scales.

The rest of the paper is organized as follows. In section II we present the general ECDP architecture. In section III we present the methods ECDP uses to calculate the rates and prices. ECDP content source selection mechanisms are presented in section IV. In section V we present the content index management component of ECDP. In section VI we show how ECDP content index management scales with the growth of the number of content records. Sections VII and VIII present the schemes ECDP uses to enforce the rate allocations and bandwidth prices without having to change the TCP/IP stack and packet header formats. The summary of the ECDP algorithm is discussed in section IX. We discuss ECDP deployment scenarios in section X. We evaluate the performance of ECDP in section XI. Analysis of related work is given in section XII. Finally we give conclusion of the paper in section XIII.

II. General ECDP Architecture

In this section we give a description of the general ECDP architecture starting with the network and content (data) models.

A. Network and Content Model

The network model of ECDP consists of a graph $G = (N, E)$ of nodes N and edges E as shown in figure 2. The node set V consists of the CDN servers which provide content and the peers which provide and/or request for content. The edge set E consists of all edges going to and from the nodes. All nodes are linked with each other over the Internet which may consist of multiple backbone networks. Each node has link with specified upload and download capacities it buys (gets) from network operators. The operator backbone network usually has enough bandwidth to provide bandwidth guarantee to the users (nodes). This is usually done using protocols such as the OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol [18].

The ECDP data model consists of content which is sent from the CDN servers or from some peers and exchanged between the peer nodes. We classify the data (contents) into none real-time ordinary static file (OSC), a realtime (live and none-live) streaming video content like 2-dimensional (2D) YouTube or a 3-dimensional (3D) video content [3]. The 3D Tele-Imersive content involves multiple streams from different view angles which have to be synchronized by the receiving end to produce a 3D multi-view streaming video. To synchronize the contents, ECDP uses content lifetime threshold based on how long a receiving node can buffer. For a stringent 3D Tele-Imersive environment, where the peers have to produce interactive content, the content lifetime becomes very small to ensure a very small delay. For most cases where nodes view the 3D content, the content life time can be relaxed.

B. Logical and Physical Architectures

The ECDP architecture aims to efficiently distribute content to network peers benefiting all content actors (content providers, content users and network operators). As shown in figure 1, it consists of a content information (index) manager (CIM) and peer agent (PA). A CIM chooses content source to the requesting peers and a PA connects a peer with the CIM. The CIM is made up of the light weight front end server (FES), content information database (CID), the complaint manager (CM) and the archive manager (AM). The CID consists of a database of contents information such as the source peers, source upload rates. The CM manages reports about misbehaving peers. The AM manages old content information and transaction logs to perform offline content index analysis. The FES forwards requests to register a new peer, a new content, or requests for a content to the respective CID tables as discussed in the section V. The FES also forwards peer complaints to the CM. The CM contacts the AM for complaint history. The CID archives old content information at the AM.

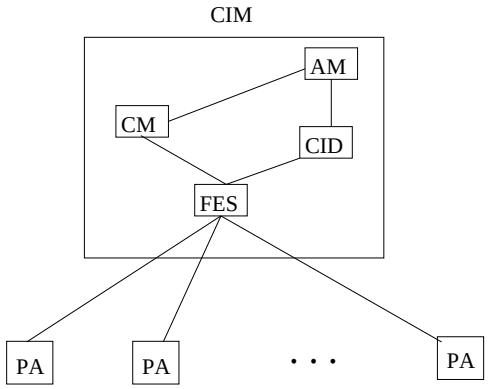


Fig. 1. The ECDP Logical Architecture

The ECDP physical architecture can be described by figure 2. The architecture consists of the peer nodes with their PA, the CIM and a big content source peer connected to its CDN with a bigger link. The big content source sends its content to the content distribution network (CDN) which informs the CIM of the new content. The content source which can be any peer with a PA can also inform the CIM of its content directly. The other peer nodes can then send a content request to the content information manager (CIM) via their peer agent (PA). The peers can get the content either from the CDN or other peers whichever gives the highest throughput to price ratio as discussed in the next section IV.

Hence, ECDP consists of three logical components namely, *prioritized max/min rate allocation, bandwidth and content pricing* and *content index management*. These components are also shown in figure 3 and discussed in the introduction section I. We show how the CIM and PA perform the prioritized rate allocation and bandwidth pricing in the next section.

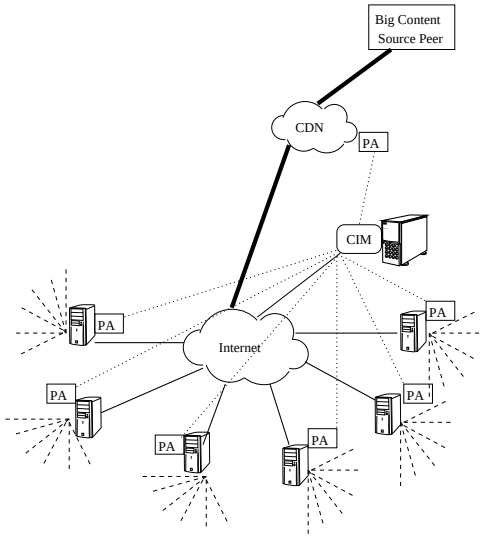


Fig. 2. The ECDP Architecture

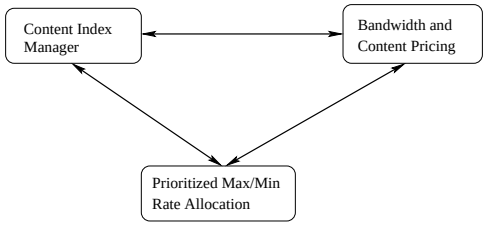


Fig. 3. ECDP Logical Components

III. ECDP Rate and Price Calculation

To obtain the rates at which each content is transmitted from one node to another node and the bandwidth usage price, ECDP first carries out temporary rate and price calculations at the CIM at every request or at every control interval τ. The rates and prices are then sent to the PAs, updated by the PA and sent back to the CIM. The CIM then uses these rate and price values to select a content source (peer or CDN server) and determine the rate at which content is transmitted.

To define the ECDP rate and price metrics, we first present the following notations in table I.

For each ECDP parameter $X \in \{R, C, Q, \hat{N}, N, n^j, R^j, M^j, p, \wp^j\}$, we use the notation,

$$X_{d,u} = \begin{cases} X_d & \text{if } X \text{ is a downlink ECDP parameter,} \\ X_u & \text{if } X \text{ is a uplink ECDP parameter.} \end{cases} \quad (1)$$

We next give short descriptions of the ECDP parameters.

A. Calculations At CIM

Whenever a node sends a request for a content to the CIM, the CIM does the following.

- Selects the best content source for the requesting peer node based on the content source selection policy discussed in section IV.

TABLE I
ECDP PARAMETERS

Variables	Description
$C_{d,u}$	Link capacity
τ	Control interval
$R_{d,u}(t)$	Base link rate allocation of the current interval (round)
$N_{d,u}(t)$	Number of flows in the link during the current round
$R^j_{d,u}(t)$	Link rate allocation of flow j for the current round
$M^j_{d,u}$	Minimum rate requirement of content flow j
$p_{d,u}(t)$	Per packet price
$\wp^j_{d,u}$	Priority weight of flow (stream or chunk) j

- Subtracts M^i_u of request i of the requesting peer from the remaining uplink capacity of the content source and M^i_d from the remaining downlink capacity of destination peer to reserve a minimum bandwidth requirement for the requesting peer. This involves only a single subtraction operation. This remaining capacity is used in equation 2 of the rate calculation. If either of the remaining bandwidths is negative, the CIM informs the requesting peer that its request cannot be fulfilled.
- Increments the flow priority weight sum used in equation 2. This involves one addition instruction. The flow priorities are globally known to the CIM or specified by each requesting peer. The PA and the CIM then calculate the corresponding weights of the priorities.
- Updates the base uplink (u) and downlink (d) rates and prices of the affected peers, using equations 2 and 4 with two division instructions each.
- After accumulating the remaining bandwidth values and the sum of the priorities used in equation 2, the calculations of the base rate using equation 2 and price values using equation 4 can be done periodically to further reduce more computational overhead.
- Sends the ID (address) of the selected content source and the uplink rate value $R_u(t)$ from the selected source to the PA of the requesting node.
- Sends the ID of the requesting node and the downlink rate $R_d(t)$ to the PA of the selected source.
  The PA then performs the tasks discussed in section III-B.
- At the CIM when the flow of the requesting peer finishes (downloading the content), the remaining uplink bandwidth of the content source and the remaining downlink bandwidth of the receiving peer are increased by the minimum rate requirement of the flow which finished and the priority respective weights sums decrease by the priority weight of the flow which finished.
- The total amount $\ddot{E}$ of credit, the content source earns, and the total amount $\ddot{P}$, the receiving peer pays, each increases by the $contentSize \times p_{d,u}(t)$.
- The base rate and price values are then updated accordingly using equations 2 and 4.

The temporary down-link (d) and up-link (u) rates of every node (peer or CDN server) are calculated by the CIM as

$$R_{d,u}(t) = \frac{C_{d,u} - \sum_j^{N_{d,u}} M^j_{d,u}}{\sum_j^{N_{d,u}} \wp^j_{d,u}} \tag{2}$$

where the notations are described in table I and $\wp^j_{d,u}$ is the priority weight of request j. If all requests have the same priority, $\sum_j^{N_{d,u}} \wp^j_{d,u} = N_{d,u}$.

The temporary uplink and downlink rates R^i_u and R^i_d of flow i are given by

$$R^i_{d,u} = M^i_{d,u} + \wp^i R_{d,u}(t). \tag{3}$$

The temporary per packet prices for the uplink (u) and downlink (d) are calculated as

$$p_{d,u}(t) = \frac{p_{d,u}(t-d) \times R_{d,u}(t-\tau)}{R_{d,u}(t)} \tag{4}$$

where the notations are also described in table I.

When a request for content is made, the temporary rate and price calculations ensure that the CIM does not result in assigning requests to peers they do not have enough resources for, leaving the refined distributed rate and price calculations to the peers as shown in section III-B below.

B. Calculations At PA

When the CIM assigns stream requests to nodes selected by a content source selection algorithm discussed in section IV, it also sends the new rate values obtained by equation 2 to the content source and destination PAs as discussed in section III-A.

When a PA receives the rates $R_{d,u}$ of its uplink and downlink flows from its CIM, it performs the following.

- Uses the uplink and downlink rate values of each of the flows of its node received from its CIM to obtain the effective flow count for all uplink and downlink flows of its node using equations 6 and 5.
- Calculates a new rate value using the effective flow count as given by equation 7. This new rate ensures that a capacity unused by some flows is being used by other flows making ECDP an max-min fair algorithm. This is because some uplink flows may be bottlenecked at the downlink and vice-versa.
- Calculates the new price value based on the new rate values using equation 4.
- Sends the new base rate values obtained using equation 7 back to the CIM. The new price values can also be sent to the CIM saving the CIM some computational costs.
  The CIM then calculates its new price values and uses both the new rate and price values to select content sources (peers or CDN servers) for each request for content.
- Calculates the new rate $R^i_{d,u}$ values of each of its uplink and downlink flows (streams) i using equation 8.
- Enforces this rate allocation using schemes discussed in section VII.

With the temporary uplink rate of a flow k from a content source as R^k_u and the temporary downlink rate of the flow to the destination by R^k_d both obtained using equation 3, if $R^k_u > R^k_d$, then the content source of flow k should not send at the rate of R^k_u for flow k as it is bottlenecked in the last

link to the destination. On the other hand if $R_u^k < R_d^k$, the destination node cannot receive (download) at the rate of R_d^k for the flow k. In these cases, other flows sharing the links with flow k should be able to use the corresponding uplink or downlink bandwidth unused by flow k to ensure that ECDP is max-min fair. To do this, we previously introduced a concept, called *fractional flow* [19], where some flows which cannot use the bandwidth allocated to them are counted as partial flows or fraction of a flow. We call such a count of a flow *an effective flow count*. The effective flow count of flow k at the source node is given by

$$n_u^k = \begin{cases} \frac{R_u^k}{R_d^k} & \text{if } R_d^k > R_u^k, \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

The effective flow count of flow k at the destination node is given by

$$n_d^k = \begin{cases} \frac{R_d^k}{R_u^k} & \text{if } R_u^k > R_d^k, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

Each PA then obtains new uplink and downlink base rate values as

$$R_{d,u}(t) = \frac{C_{d,u} - \sum_j^{N_{d,u}} M_{d,u}^j}{\sum_j^{N_{d,u}} \wp_{d,u}^j n_{d,u}^j}. \tag{7}$$

The new per packet prices for the uplink and downlink of a node are then obtained using equation 4.

Besides, a node resets the up and downlink rates of its' flow i as

$$R_{d,u}^i = M_{d,u}^i + n_{d,u}^i \wp^i R_{d,u}(t). \tag{8}$$

Equivalently, the uplink rate R_u^i of the flow i at a node can also be calculated as

$$R_u^i = M_u^i + n_u^i \wp^i R_u(t). \tag{9}$$

If there is a CIM which knows every request for content and every end of a flow, then equations 6, 5, 7 and 4 can also be calculated using the CIM itself. However, in the case of different CIMs being responsible to different groups of nodes for instance based on locality, then the content source selection by one CIM at a given level of the CIM hierarchy may not be known to other CIMs. In this case the new rate and price values should be obtained by the PA of each node.

C. Incentives in Free Content Delivery

ECDP can also operate under a free content delivery mode, where the monetary amounts to be earned and paid can be converted into downlink bandwidth incentives as follows.

- First, a requesting peer sends its current base downlink rate $R_d(t)$ along with the other fields to the CIM.
- The CIM then selects a content source with a base upload rate of $R_u(t)$ to the requested content based on the specified policy.
- The CIM informs the content source to rate-limit the requesting peer at a base rate of

  $$\ddot{R} = \ddot{w}(\ddot{E}, \ddot{P}) \times min(R_d(t), R_u(t))$$

  where $\ddot{w}(\ddot{E}, \ddot{P})$ is the weight function of the total monetary amount $\ddot{E}$ the requesting peer has earned and the total amount $\ddot{P}$ the peer has to pay. The *min* is a minimum function. In this study we set

  $$\ddot{w}(\ddot{E}, \ddot{P}) = \frac{\ddot{E}}{\ddot{P}}. \tag{10}$$

  Other pricing and weight functions can also be used in ECDP.
- The new weights $\tilde{\wp}_u^j$ of every request j from the requesting peer is then set as $\tilde{\wp}_u^j = \wp_u^j \ddot{w}(\ddot{E}, \ddot{P})$. This new weight is the product of the peer weight, $\ddot{w}(\ddot{E}, \ddot{P})$, and the flow (stream) priority weight, $\wp_u^j$.
- CIM obtains the rate allocation of the request j made by the peer as $\ddot{R}^j = M_u^j + \tilde{\wp}_u^j \ddot{R}$.
- CIM also estimates the content transfer duration of the flow of request j as $estContentDuration = \frac{contentSize}{\ddot{R}_j}$ where *contentSize* is given in the *tblRequestedContent* of the CID. This makes sure that the content source losses bandwidth credit if it does not enforce the rate limit request by the CIM and gives higher rate to the requesting peer.

D. Content Lifetime

When a content is produced by a source node, the source attaches the creation *timestamp*. The CIM maintains a content lifetime threshold Ψ. For realtime applications such as 2D and 3D streaming which are sensitive to content lifetime, Ψ is set to the desired value. In our experiments for viewers of 3D streaming [3], we set $\Psi = 2.5\,sec$. For interactive realtime applications, Ψ can be to a much smaller value. Each node in ECDP can also maintain its own value of Ψ based on its storage and downlink capacities. If the Ψ of a content expires, the CIM puts the content information to its AM. The CIM continues to find sources with a fresh (streaming) content for requests.

Each node which requests for a content maintains a buffer worth at least $\tilde{\Psi} = \Psi + RTT$ seconds for each content flow it requests where RTT is the estimate of the round trip time of the flow (stream). This is specially useful if a peer has to perform 3D display of contents. In this case to maintain synchronization between frames of different streams (flows), a node keeps a buffer worth $\tilde{\Psi}$ of early arriving frames of each flow. The longest a node waits before it renders streams is then $\tilde{\Psi}$. In the results reported in this paper we have used the *hop count* field with the maximum RTT value of 500 ms per hop to estimate the Ψ for the streaming experiments.

E. ECDP is TCP friendly

Theorem 1: An ECDP rate allocation of a flow which is not bottlenecked at a link l is TCP friendly to all flows sharing link l.

Proof: If a flow i is not bottlenecked at link l, it cannot congest link l regardless how much its sending rate increases. This is because the flow i has another bottleneck which limits its sending rate. This in turn means that TCP flows sharing link l with flow i have enough bandwidth at link l to use. This implies that TCP fairness is not an issue at link l and flow i is TCP friendly. ■

Even though ECDP handles scenarios where the bottleneck link can be somewhere in the backbone network, the bottleneck link in the ECDP architecture is usually going to be at the last mile links to and from the peers. This is because (1) users (peers) usually buy a guaranteed bandwidth and (2) the peers which can use a specific peer as a source of their content are usually scattered over a wide area each using different paths in the backbone network. Hence, if the ECDP flows are not bottlenecked at a link which they share with TCP, then they are TCP friendly based on the above theorem 1. In a scenario where the bottleneck link is in the backbone network, the ECDP flows will drop or delay packets. This congestion signal can be detected by the PA of each peer which counts the number of successfully transmitted packets. The PA compares this count over a time interval against the minimum of the uplink and downlink rates of the flow. If the PA finds that the backbone network is congested, it uses the maximum congestion window and receive window to enforce the rate allocations as discussed in section VII.

F. When a Flow Ends

When a peer wants to end a flow (stream) due to for instance 3D view change, the node sends the contentID of the flow (stream) it needs to end. The CIM then finds the corresponding global contentID in its content table, removes the contentID and releases the associated resources. It then updates the corresponding content source and destination rate and price values. The CIM also finds a new content source to all other peers which are actively downloading the content from the peer which wants to end it. Here, the CIM uses the original content source as the new content source for the peers which are using the content whose source is ending it. This is because if a new peer (which is not the original content source) is chosen to be the new source of the content, it is difficult to find (trace) out whether one of the parents (ancestors) of this chosen peer is the peer which is ending the content or not.

After a peer completes downloading a (non-streaming) content, it requests for the decryption key. The CIM then updates the rate calculations of the content source and destination based on the equations in sections III-A and III-B.

In this section, we have been discussing how the CIM and PA perform the rate allocation and bandwidth calculations in a TCP friendly manner. In the next section we discuss mechanisms which the CIM with the assistance of the PAs uses to select a content source for a requesting peer.

IV. Content Source Selection

Once the CIM receives the new rate values from each PA, it obtains the new price values using equation 4. Then any of the following policies can be used for selecting a content source for the requesting nodes. The source can be another peer node or a CDN node.

A. Highest Rate to Price Ratio Policy (HRPR)

In this HRPR policy, the CIM keeps the ratio

$$K_{d,u}(t) = R_{d,u}(t)/p_{d,u}(t) \tag{11}$$

of the rates to their respective prices in its peer table as shown in section V-A1. When a node requests for a content, the CIM chooses a content source which gives the highest value of $K_{d,u}(t)$. This approach enables the CIM to choose a node which gives the highest rate with the lowest price. In the analysis of this paper we use this content source selection strategy.

B. Highest Rate Policy

In this policy the CIM selects a content source which provides the highest rate to each node irrespective of the price. So a node which is allowed to download from a content source with the highest rate pays the corresponding price. The nodes can also earn credit by allowing other nodes to download from them and then get a service whose price is equivalent to the credit they have earned as discussed in section III-C.

C. Best Rate Fit Policy

When a node requests the CIM for a content, the CIM can also choose a content source whose upload rate is the smallest value greater than the download rate of the requesting node. This approach allows the CIM to do a best fit allocation to allow big upload requests.

D. Smallest Price Policy

If a node which requests for a content doesn't want to pay more or doesn't want to spend more of its credit, it can request a smallest price policy. In this case, the CIM chooses a content source with an upload value of at least as much as the minimum required rate for the content and with the smallest price.

E. Lowest Latency (Local Network) Policy

A user's request may have some latency constraints. In this case a user may request a node with the shortest latency. To deal with this scenario, we group peers with similar IP prefixes together. This can be done by hashing the most-significant bit-group in the IP address of the content request packets of the registering peers. We can then have one CIM responsible for each group of users (peer nodes) forming a hierarchical structure of content information managers as shown in figure 4. This policy can have a significant advantage in reducing backbone network link congestion as many requests can be served locally. This is another benefit to network operators. Besides, users in the same geographical location may tend to have interest to the same content making it easy for the content source selection algorithm to decide.

To use this policy, users send a request to the FES of the CIM which then hashes the requester's IP prefix values and

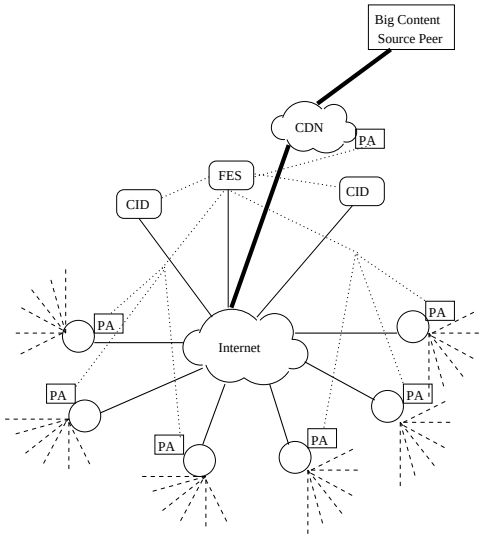


Fig. 4. The ECDP Architecture

forwards them to their respective CID tables. This approach also allows ECDP to scale as discussed in section VI.

F. Small Content Lifetime (Hop Count) Policy

The peers in the ECDP have a strong incentive to store and share the contents they download. As every upload can result in credit which can translate to monetary rewards or high download rate. A node can also inform the content index manager (CIM) that it does not want to serve a specific content. Besides, a peer which has big enough buffer can store early arriving streams to create a 3D tele-Imersive [20] view along with other streams which arrive late. In a scenario where a significant number of peers have limited buffer, ECDP can either use a small content lifetime as discussed in III-D or follow a *small content lifetime* policy. In this policy, the CIM uses a content hop count field in its content index database (CID) along with locality information. Here a content source with the lowest hop count is selected to serve the requesting peers. The CIM first tries to find such content in the local CID. If the content with the desired hop count cannot be found in the local CID, it is searched in the master *tblSelectedSource* table as shown in section VI-A. If such content with the desired hop count cannot be found, the default *highest rate to price ratio* policy discussed in section IV-A is used.

G. Private Group Policy

This ECDP policy allows content to be shared within a specific group of peers which can be social or organizational groups. Each private group can form its own CIM with any of the above server selection policies. This can enable ECDP to deploy Facebook like applications such as the Diaspora [5], [6]. A distributed network of CIMs can also be formed where CIMs exchange public content information based on privacy settings in an adhoc or hierarchical manner. Any peer can then subscribe to different CIMs for different contents forming a distributed content networking.

So far we have been discussing the two major components of ECDP which deal with the prioritized rate allocation and resource pricing. After presenting mechanisms of how the uplink and downlink rates for each peer and the corresponding bandwidth prices are calculated by the CIM and PA we have also discussed how the CIM uses these metrics to select a content source for a requesting peer. We next present an efficient content index management scheme which the CIM uses to select the best content source for a requesting peer.

V. Content Index Management

In ECDP some peers or content providers provide content by registering their content information at the content index manager (CIM). Other peers request the CIM for a specific content. In this section we show how such contents are registered, requested and their source selected.

A. Content Index Database

The registered content information is stored at the content index database (CID) which is part of the CIM system. The CID consists of the *tblPeer*, *tblContent*, *tblSelectedSource*, *tblRequestedContent* tables as shown in figure 5. The *tblPeer-Content* table is used to link the *tblPeer* and *tblContent* in a many-to-many relationship.

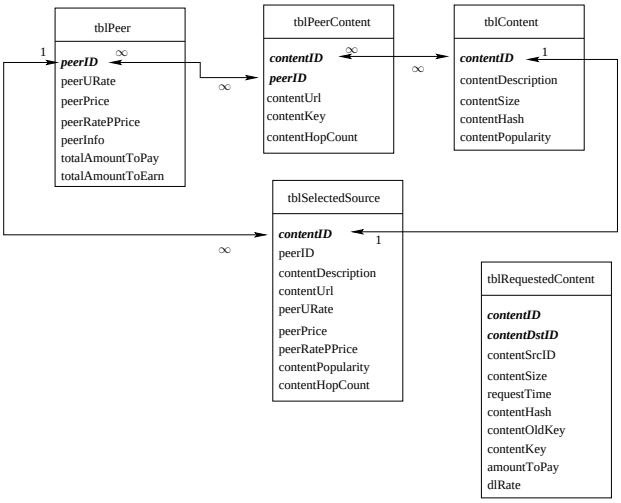


Fig. 5. The CID Architecture

1) Peer Table: The *tblPeer* contains the fields described in table II. Initial content providers need to fill in all the fields of this table. The *peerInfo* contains real content provider information such as telephone number, address and/or credit card number. Such confirmed information holds each content provider accountable for the nature of the content provided. The *peerInfo* field is also used by the content providers to charge peers for none-free contents. Once a peer receives a content, the CID registers the peer as having the content unless the peer indicates that it does not want to serve the content. The peers which are not the original sources of the content do not have to provide their *peerInfo* unless they want to receive monetary value of the credit they earn. The *peerID* field is

the primary key of the *tblPeer*. It is preferred to be the IP address of the peer. The peers have incentives to provide their correct IP addresses. This is because if a peer gives a wrong IP address, it can not get a content as a source sends its content to an IP address it obtains from the CIM.

TABLE II
PEER TABLE FIELDS

Field name	Description
peerID	Unique peer identifier
peerURate	Current base uplink rate, $R_u(t)$ of a peer calculated using equation 2
peerPrice	Per unit uplink cost of a peer node
peerRatePPrice	Peer rate per price calculated using equation 11
peerInfo	Real content provider information
totalAmountToPay	Total monetary amount a peer needs to pay for downloading a content
totalAmountToEarn	Total monetary amount a peer earns for uploading a content

If the peer is just joining the CIM, its uplink rate, *peerURate*, is the the total uplink capacity it uses to earn credit from other peers to which it uploads content. The peer has an incentive to dedicate more uplink capacity, as more uplink capacity can bring the peer more credit (monetary values). After the initial calculation by the CIM using equation 2, *peerURate* is updated by each peer using equation 7. To minimize the computation load of the CID, the *peerURate* can also be entirely calculated by the peers in a distributed manner and sent to the CIM every control interval τ. If the peers send their download rates to the CID and if there is enough server processing (computation) capacity at the CIM, all rate computations given by equations 2 and 7 can also be done by the CIM servers in a centralized manner. In this paper we use the approach where initial simple rate computation is done by the CIM servers and the more detailed rate update computation is done by the peers. The peers then send the update to the CIM servers.

The *peerPrice* in our study is per packet cost where one packet in this study is 1000 Bytes. The *peerPrice* is initially set to be the unit content cost plus basic initial user defined link cost. The content cost is zero for a free content scenario and the initial link cost in our study is determined by the CIM system. After the initial cost, *peerPrice* is calculated adaptively by the CID servers using equation 4 for each link.

The default content source selection policy we use in this study is the *Highest rate to Price Ratio Policy* discussed in section IV-A. To implement this policy the *tblPeer* maintains the peer rate per price *peerRatePPrice* field.

The *amountToPay* and *amountToEarn* fields are updated by the *tblRequestedContent* table. A peer gets an additional amount in dollars for each content it serves and pays a certain amount for each content it downloads.

2) Content Information Table: The second table the CID keeps is the content information table which we call *tblContent* in this paper. This table contains the fields shown in table III.

The *contentSize* is used by the CIM to charge the peer which receives the content. The CID uses this content size to obtain the per content amount a peer has to pay. The *contentHash* is used by the content receiving peer to check for content integrity. Every time a content is selected by a peer, the popularity of the content increases.

TABLE III
CONTENT INFORMATION TABLE FIELDS

Field name	Description
contentID	Uniquely identifies content chunk or stream in the CIM
contentDescription	A textual description of the content
contentSize	Size of the content in KB
contentHash	To check for content integrity
contentPopularity	The number of times a content with *contentID* is requested

3) Peer-Content Linking Table: This *tblPeerContent* links the *tblPeer* with the *tblContent* in a many-to-many relationship. To achieve this *tblPeerContent* consists of the primary keys *peerID* and *contentID* of *tblPeer* and *tblContent* tables respectively. The table also contains the peer specific fields, *contentUrl*, *contentKey* and *contentHopCount*. The current location of the content in a peer with *peerID* is *contentUrl*. The source peer encrypts its content with the symmetric key *contentKey*. After a peer receives a content from another peer or from a the original content server, it requests the CID (*tblRequestedContent*) for the key to decrypt the content. The *contentHopCount* is set to 1 if the peer is the original content source. Every other peer which receives the content increments the value of the field by 1. This field along with locality information for instance helps estimate the streaming content age since its initial distribution.

4) Selected Source Table: From all the original content servers and peers which have a specific content, a source for a requested content is selected based on the content source selection policy discussed in section IV above. For each content source selection policy a table called *tblSelectedSource* is produced by a query from the *tblPeer*, *tblPeerContent* and *tblContent* tables. For the *Highest Rate to Price Ratio Policy* (HRPR) used in this paper, the *tblSelectedSource* has the fields, *contentID*, *peerID*, *contentDescription*, *contentUrl*, *peerURate*, *peerPrice peerRatePPrice* and *contentPopularity*. This table can be sorted in descending order of popularity to put the most popular contents at the top even though every content can be looked up in constant time.

5) Requested Content table: The requested content table, *tblRequestedContent*, consists of the the fields, *contentID*, *contentSize*, *contentSrcID*, *contentDstID*, *requestTime*, *contenOldtKey*, *contentKey*, *amountToPay* and *dlRate*. The *contentID* and *contentSize* fields correspond to the the requested content. The *contentSrcID* field is the *peerID* of the peer or server which is selected to serve the content. The *contentDstID* field is the *peerID* of the content requesting peer. The field, *requesTime* is the time when a request for the specific content was made. The *contenOldtKey* field is a symmetric key with which the content was encrypted and by which the content receiver will decrypt the content. Once a peer with *contentDstID* requests for this key to decrypt the content it downloaded, its *amountToPay* value is set to the product of the *contentSize* and the content duration $\check{D}$. Here $\check{D}$ is the difference of the current time (content delivery time) and the content *requestTime*. The *totalAmountToPay* of the peer with

contentDstID and the *totalAmountToEarn* of the peer or server with *contentSrcID* that serves the content each increase by *amountToPay*. The *contentKey* field is a new symmetric key generated for the content by a specific peer generated by the CID. The content requesting peer uses this key to encrypt the content when selected by the CIM to serve the content. Once the *contenOldtKey* is successfully received by the peer which requested the content, and after other tables of *contentSrcID* and *contentDstID* are updated, the record entry of these fields in *tblRequestedContent* is deleted. The *dlRate* field is set to the minimum of the downlink (to the destination) and uplink (from the source) rates of the requested content.

In the next section we discuss how the CID tables scale with the growth in the number of content and peer record entities.

VI. Scaling User and Transaction Management

In this section we discuss how ECDP scales to an increase in the number of users and with the multiple variations in the request arrival and completion patterns. The CID tables can be scaled with increasing number of peers and contents by using multiple data center like servers along with appropriate hash functions. If the number of servers available for the *tblPeer* table is S_p, $sigBits(peerID)$ gives the integral value corresponding to the most significant bits of the peerID field. How many significant bits of the peerIDs we take depends on how many content entries we have. Taking fewer significant bits for instance means we need fewer servers (smaller S_p) as more peerIDs can be mapped to a single server. A record for *peerID* goes to *tblPeer* located at server $sigBits(peerID) \bmod S_p$. Here the servers are identified by positive integral values and mod is the modulo operation. A record for *contentID* of *peerID* goes to *tblContent* located at server $sigBits(peerID) \bmod S_p$. This ensures that the content and peer information are located in the same server for easier local look-up.

Such hashing by *sigBits(peerID)* helps that content information of peers whose IP addresses have the same domain go to the same server. In this case if a peer in one index server is selected by the CID as a source of a content to another peer in the same index server, then the content source selection strategy becomes local. Such local content source selection mechanism can help peers achieve low download latency as the content can be served from another peer in their local network.

When the request arrival and completion vary so much, the PA needs to recompute the rate given by equation 7 multiple times. Furthermore the PA needs to update the rate values at the respective CIM. Since each CIM obtains temporary rates using equation 2, the PA does not have to send every update to the CIM. The PA can send updates every user-defined control intervals.

Equation 2 used by the CIM only needs one subtraction (addition) and one division per new flow request arrival or departure to obtain a temporary uplink rate for each peer. It also needs one multiplication and one division to obtain the temporary price given by equation 11. Since the process is adaptive, some CIM rate and price updates can as well be skipped as they can be updated by the rate the PA send for each of their links. The CIMs using equation 2 also do not need to obtain the temporary downlink rates and prices the downlink rates and prices can be sent by the content requesting peer. In these cases the CIM only needs to check if the selected source has enough remaining upload link capacity to satisfy a minimum rate requirement $M^j_{d,u}$ of the request j.

A. Database Partition and Aggregate

Assigning *tblPeer* and *tblContent* tables to different index servers based on the peer ID (IP) essentially partitions the CID into multiple local databases. Each local database matches content source and destination peers located in the same network domain and local area. We call such content matching a *local content source selection strategy*.

If content cannot be found in a local network or if peers in other local networks can provide a higher upload rate and lower price, then the source selected to serve a content can be from a different network domain, different area or even different country. Such a content source selection strategy where a content source can be chosen from a different network domain (area) is called *global content source selection*.

To achieve global content source selection, the CID needs to know a source with the highest upload rate and lowest per packet price for the requested content. The CIM achieves this by using a map/reduce [21] like framework as shown in figure 6. For the content source selection strategy we use in this paper, each local CID database *tblPeerContent* is sorted in descending orders by *contentPopularity* and then *peerRatePPrice* for each content. So here we have each *tblPeerContent* information *mapped* to many local index servers (many CIDs).

For each content, a record with the highest *peerRatePPrice* among all the *tblContent* tables in each local CID database is selected *(reduced)* into the *tblSelectedSource* table and placed in another index server. This is like the reduction phase in the map/reduce framework where the *maximum of* is the reduce function. Each CID continuously sorts the *tblPeerContent* table by the *peerRatePPrice* field for each content with the changes in the upload rates and prices of the corresponding peer.

If the value of *peerRatePPrice* field in a *tblPeer* table changes, first, each *contentID* content of the *peerID* peer in the *tblPeerContent* is sorted in descending order of *peerRatePPrice*. Then for each content of *peerID* in the *tblSelectedSource* table, if the highest *peerRatePPrice* of *peerID* is higher than the *peerRatePPrice* of the corresponding content in *tblSelectedSource*, then the values of the *peerID* and *peerRatePPrice* fields in the *tblSelectedSource* table are replaced with the corresponding values in the *tblPeer*. The *tblSelectedSource* table is sorted by *peerID*. Hence all contents of the *peerID* field are located once the first content of *peerID* is found. Such a procedure of constantly updating the *tblSelectedSource* table ensures that the table always consists of the list of contents given by the peers with the highest upload rate and lowest price (highest rate to price ratio).

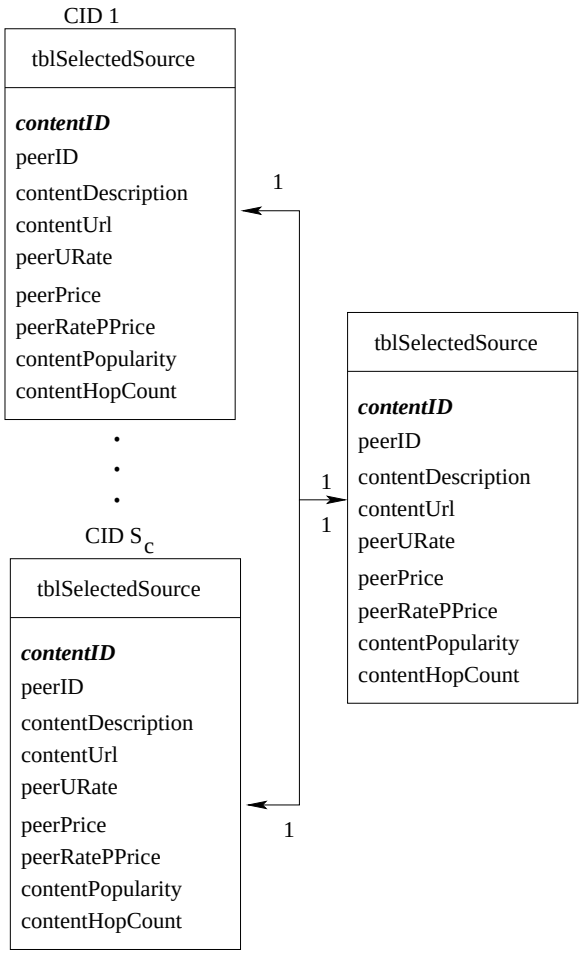


Fig. 6. The CID Partition and Aggregation

B. CID Complexity Analysis

The CID operation of adding new peers to the *tblPeer* is of constant order $\mathrm{O}(1)$ as the *tblPeer* does not have to be sorted out. Each content in the *tblPeerContent* has to be sorted out by *peerRatePPrice*. Hence adding a new content entry to the *tblPeerContent* table has a complexity of order $\mathrm{O}(log(N_c))$ where N_c is the number of peers which have the same content.

Whenever a peer gets a content that it requested, then (1) the uplink rate of the content source decreases according equation 2 and (2) the peer which gets the content becomes one of the content sources. These two operations require two $\mathrm{O}(log(N_c))$ operations for each content. As the *tblPeerContent* in each of the CID partition is sorted, updating the *tblSelectedSource* for each of its contents is of constant order.

When *tblRequestedContent* is updated upon a successful download of a content by a peer, the corresponding values of *peerRatePPrice* and *totalAmountToEarn* of a source peer or server and the *totalAmountToPay* of a receiving peer are updated in constant time by using the matching *peerID* field.

In the above sections we have discussed how the bandwidth rates and prices are calculated and how content and peer information are managed using databases. We next show how the rates and prices are enforced so as to increase content transfer throughput and decrease content transfer delay.

VII. Enforcing Rate Allocation

When nodes request a certain content, the CIM selects the content source with the associated uplink rate values. It also informs the content source of the downlink rate of the requesting node. With the content source having its uplink rate and the downlink rate of the receiving node any of the following schemes can be used by the CIM to enforce the rate allocation.

A. Enough Backbone Bandwidth

The backbone links in the Internet which the nodes use and which are represented by the "Internet" node in figure 2 are not usually congested as can also be seen from [22]. Each user of the ECDP mechanism can also have bandwidth service level agreement from the operators which guarantee the desired capacity. Under this scenario the only bottleneck links are the last links to and from the ECDP peer nodes. Hence the sources of the desired contents can set their congestion window sizes (*cwnd*) to the product of their uplink rate value calculated using equation 8 and their round trip time (RTT) as follows.

First the destination node sets its receive window w_r^i of flow i as

$$w_r^i = R_d^i(t)RTT^i. \tag{12}$$

Then the corresponding source of the flow (stream) i sets its congestion window w_i as

$$w_i = min(w_r^i, R_u^i(t)RTT^i). \tag{13}$$

B. Scarce Backbone Bandwidth

If the bottleneck link is somewhere in the Internet which is described as "Internet" node in figure 2, then the destination of flow i sets its receive window size as given by equation 12. And the source of flow i sets its maximum congestion window size w_M^i as

$$w_M^i = R_u^i(t)RTT^i. \tag{14}$$

A node can detect whether or not the bottleneck is in the link other than the last links to and from the source and destination peers using different ways. For instance if a packet loss is observed for flow i after the rate is enforced using equation 13, then the TCP source of flow i can assume the bottleneck link is other than the last links to/from the source and destination nodes. The PA of the receiving end can also count the number of received packets (bytes) per unit time to obtain the actual download rate per content. Similarly the PA of the content source can also estimate its uplink rate of a specific content by counting the number of successfully acknowledged packets (bytes) per unit time. The PA of the source and destination of the content then report this rate to the CIM per a specific content. The CIM then replaces the *peerURate* of the content in *tblSelectedSource* with the minimum of these two values. The source and destination peer also update their rate calculations using equations 5, 6 and 7 to re-allocate unused capacities to other requests.

C. Packet scheduling

In cases where the requested content is a video stream, the source of the content can stream its frames by scheduling them at $\frac{1}{R_u^i}$ apart where R_u^i is given by equation 8.

D. Peers have no incentive to not obey the ECDP mechanism

In this section we discuss possible misbehaving scenarios and show that no content source or content destination in ECDP has an incentive to misbehave. In fact the credit misbehaving sources earn can be revoked by the CIM.

The first misbehaving scenario is advertising a bandwidth one does not have. The only reason a peer can advertise higher uplink bandwidth than it have is to attract more customers (content requesters). However this mechanism does not result in higher credit as the credit (monetary amount) is earned based on the amount of data a source uploads per unit time. If content source attracts more peers than its bandwidth can handle, then it takes longer to serve each of these requesters. This results in the peer taking longer to earn its credits. Besides, if the PA of the peers report the low rate and if the CIM confirms it using the fields in the *tblRequestedContent*, the misbehaving source (a source which advertises a higher than available bandwidth) gets its credit for a specific content revoked. The *tblRequestedContent* of the CID has the *requestTime* and *contentSize* fields. When a peer receives a content, it requests the CID for the decryption key. The CID of the CIM can then compare the $actualTime = currentTime - requestTime$ against $promisedTime = contentSize\,/\,dlRate$, where $currentTime$ is the time when the request for the decryption key arrives at the CIM and $dlRate$ is the minimum of the uplink rate and downlink rate of the requested content. If the $actualTime > promisedTime + toleranceVal$, then the CIM via its CM concludes that the source is not uploading at the rate it suggested. Here $toleranceVal$ is a user-defined tolerance value.

VIII. Enforcing Prices

If a peer chooses a *monetary incentive* mode of ECDP, it registers itself at the CIM providing more detailed *peerInfo* with which it can be charged or credited. A peer which chooses a *bandwidth incentive* mode of ECDP first registers and accumulates enough credit by downloading popular contents chosen for him by the CIM. The initial contents which such peer downloads in the *bandwidth incentive* mode are randomly chosen by the CIM to avoid a free loading scenario where peers download the content they want for free and then disappear. After the initial content assignment by the CIM, and after the peer accumulates enough credit to download a content, then it can download a content it chooses. If peers are willing to have a pre-paid account or provide information (like phone numbers or credit card) by which they are held accountable for the contents they download, they are not forced to download a content they do not want.

Every peer which downloads a content is charged by the CIM (its credit balance calculated) as soon as it downloads the content. The CIM charges the peer when it receives the peer's request for the content decryption key. After charging the peer, the CIM then sends the content decryption key to the requesting peer.

The next section gives a brief summary of the entire ECDP protocol which consists of multiple algorithms.

IX. Summary of the ECDP Protocol

Initialization steps:

- Each participating peer and CDN server first initializes its up link and down link base rates to the uplink and downlink capacities it dedicates to the ECDP system.
- Each participating peer and CDN server also initializes its unit per packet price (bandwidth) to some value. In this study the CIM sets the initial per packet bandwidth prices to the peers and real CDN bandwidth prices used by the AmazonCLoudFront. Even though we considered only bandwidth price in this paper, the price may include other costs such as peer storage, energy, content cost and other costs.
- Each participating peer and CDN server with a content then sends these rate and price values along with the other source and content fields in *tblPeer* and *tblContent* discussed in section V to the CIM.

Content request steps:

- Peer which is interested in a specific content (flow or stream) sends a content request to the CIM via its peer agent (PA). The most popular content information can be displayed by the CIM for other peers to see. Peers can also lookup the content from the CIM tables.
- CIM first authenticates and registers the requesting peer.
- If no peer has a desired content, the CIM sends the IP address of a CDN (cloud) server which has the content to the requesting peer and the IP address of the requesting peer to the selected CDN server.
- If there is (are) other peers which have a content requested by another peer, the CIM chooses the node (peer or CDN server) which gives the best metric based on the content source selection policies discussed in section IV.
- CIM sends the IP of the selected content source along with the base upload rate and the *contentHash* field discussed in section V to the requesting peer.
- CIM sends the base download rate of the requesting node to the selected source.
- Requesting peer downloads the content from the source whose IP address it got from the CIM.
- Both content source and destination enforce the rate allocation as discussed in section VII and VIII.
- Requesting peer gets the *contentOldKey* from *tblRequestedContent* table discussed in section V to decrypt the content it downloaded.

CIM update steps:

- CIM records the *contentID* of the content being downloaded by a peer along with the *peerID* and other fields discussed in section V into *tblPeerContent* table.

- CIM calculates the temporary upload and download base rates of all nodes based on equation 2 and send these new rates to the PA of each node (peer or server).
- CIM also calculates the temporary per packet price per uplink and downlink of each node.
- CIM sends the temporary uplink and downlink rate values to the corresponding PA of each node.

PA update steps:

- PA then recalculates the uplink and downlink base rates along with the corresponding per packet prices using equations 7 and 4.
- PA sends the updated rates back to its respective CIM and the CIM updates its *tblPeer* table.

We have designed ECDP in such a ways that it can be deployed in current networks. We next discuss scenarios where ECDP can be deployed.

X. ECDP Deployment Scenarios

One of ECDP deployment scenarios is with each peer using a personal web (content) server similar to the Diaspora social network. The personal web server can be hosted at a home server, at a friend server or at an ISP. ECDP can also be implemented in big content searching companies such as Google. This can be done by using the CIM to select the best server to serve a content. For instance when a Google customer requests for a content, the CIM finds the server with the best upload rate to the server which is the closest (best metric) to the customer. The server closer to the customer then caches the content and content information. So each Google server which is connected to the Google backbone network can have a PA which communicates with the CIM for rate and price updates.

The PA of ECDP can also have CIM like functionality where a peer (server on behalf of peer) which requests for a content first checks its local CIM database for a content. If the content is not located locally then the next level CIM can be contacted for a content. ECDP can be bootstrapped by leveraging existing search engines. When a user requests the CIM for a content and if the CIM cannot find the content entry in its CID, it then forwards the request to the Bing or Google search engine. The CIM selects the best search engine results for the requesting peer. When the peer clicks (selects) a specific content entry, CIM records that content information as being stored in the requesting peer. Next time other peers request for the same content, the CIM selects the peer which used the search engine as the content source. This can gradually lead to a decentralized Internet where contents are owned by distributed peers and not centralized entities. This can significantly decrease the backbone bandwidth congestion as more contents can be served in local networks.

Once users maintain content ownership, they can host controlled and paid advertisements (ads) in their personal web servers based on the popularity, freshness and other metric values of the content. If a content is infested with too much paid ads, other peers will select a content source with less or no ads. This can help control the number of ads peers want to make money from. The CIM can also have its own policy on the nature of contents and ads. Such a distributed nature of content distribution can also result in a reduced cost of advertisements as there is more competition due to popular contents being owned by more people in the distributed network. Hence as more users (customers) get efficient and fair incentives to participate in the content distribution, content providers, network operators and advertisers can also benefit greatly. Such a scheme where all players benefit is a stable and sustainable system.

We have performed detailed experimental analysis to evaluate the performance of ECDP as shown in the next section.

XI. Evaluation

In this section we evaluate the performance of ECDP and all its components using simulation and real world experiments. For the simulation, we implemented ECDP in the NS2 simulation and for the real experiments we implemented ECDP using Apache SQL server [23] to manage the content indices.

A. Simulation Setup

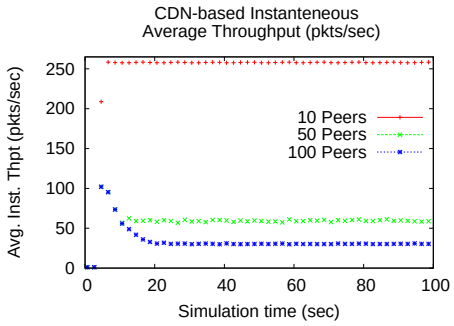
We use a simulation topology similar to the one given in figure 2. For the simulation the upload and download capacities of the links to and from the peers is 15Mbps. The link capacity to and from the CDN is $n_{peers} \times 15\,Mbps$, where n_{peers} is the number of peers. The propagation delay between the peers is taken from 4 hour PlanetLab traces [24]. The average CDN bandwidth price taken from the Amazon CloudFront [25] is $avg\_cdnPrice = \$0.176$ per GB of traffic. The initial peer bandwidth price is $avg\_cdnPrice/(2.0 \times n_{peers})$. This price adaptively increases as the peer rate decreases with more demands based on equation 4. We run different sets of experiments as shown in the following sections.

B. Pure CDN Vs ECDP-Based Schemes

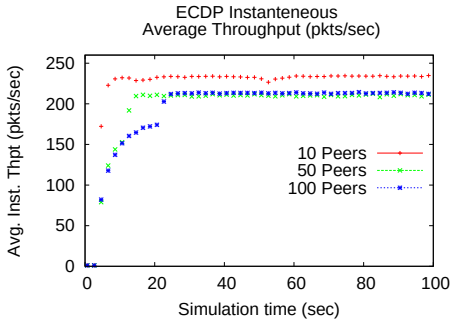
Figure 7 shows how the ECDP-based scheme scales with the growing number of content requesting peers when compared with the pure CDN-based approach. This result is consistent with detailed study [10] which shows that the hybrid CDN-P2P can significantly reduce the cost of content distribution bandwidth.

C. Pricing Evaluation

Figures 8 shows that the monetary amount spent per GB of traffic using the ECDP peering is much smaller than the fixed per GB traffic cost using the cloud CDN. The figure also demonstrates the theory behind the ECDP pricing mechanism. The few price spikes in the ECDP result when the peers download the content from the CDN cloud. This happens either because the content from the peers is too old as discussed in section III-D or none of the peers have a content source selection metric higher than that of the CDN servers based on the HRPR policy. The content provider can subsidize this bandwidth price (cover the extra bandwidth expense) as such peers which download the content directly from the source act as content seeders.



(a) Pure CDN based Approach



(b) ECDP based Approach

Fig. 7. Pure CDN Versus ECDP-based Approach

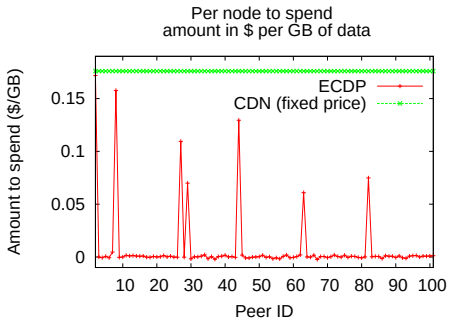


Fig. 8. Amount Spent: ECDP Vs CDN

As shown in figure 9 most of the peers do not have to pay an extra bandwidth amount to download a content as the credit amount they earn balances out the debit they incur.

D. Other P2P schemes Vs ECDP

We have also compared the performance of ECDP against other hybrid P2P and CDN schemes in terms of average chunk completion time (ACCT). Previous hybrid P2P and CDN schemes such as the Dandelion [8], PACE [14] use TCP as their transport protocol. So we show how these schemes using TCP compare against ECDP by fixing the content source selection mechanism to be the same for both.

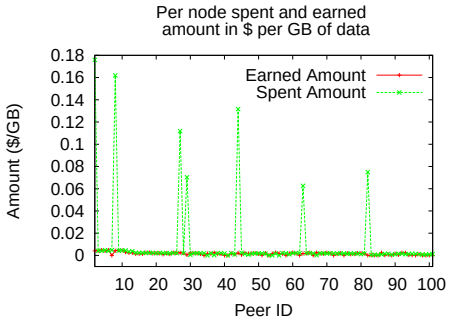


Fig. 9. ECDP Spent Vs Earned Amount

For this experiment we use 8 files with content $i, 1 \leq i \leq 8$ is with file size $500i$ KB and chunk size i is $50i$ KB. Inter-content chunk time is 0.5 seconds. All contents are requested at the same time. Each file (content) is divided into equal chunks. Content popularity is 5 for each of the contents. For the TCP-based and the ECDP approaches content destination and source are the same. For these experiments we set the minimum flow rate to 0.0 and all chunks have the same priority levels.

Figure 10 shows that the completion time of small chunks is much smaller in ECDP than the pure TCP-based approaches (PACE, Dandelion) without affecting the CCT of bigger chunks.

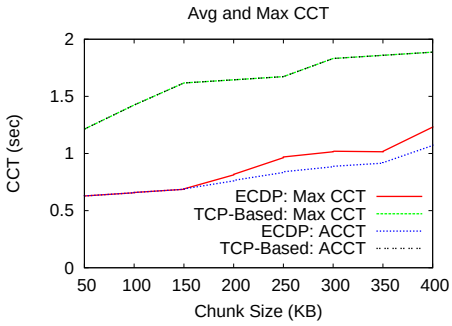


Fig. 10. Avg and Max CCT of ECDP Vs TCP-Based Approaches

E. 3D Streaming Result

For the 3D streaming experiments, we use a setup which emulates [3] with 6 streams. Each stream demands a minimum of 1Mbps capacity. Each stream i, $1 \leq i \leq 6$ has a priority weight of $1/i$. We used a content lifetime of 2.5 seconds for the streaming. So if a stream at a peer is older than 2.5 seconds, the CIM does not register the peer as having the content.

Figures 11 and 12 demonstrate the priority and minimum rate mechanisms of ECDP. As shown in the figure, stream 1 which has the highest priority weight gets highest throughput. The throughput of the other streams follows their priority weights.

Figures 13 and 14 also show how the Instantaneous throughput of the different streams evolve with time. All these plots

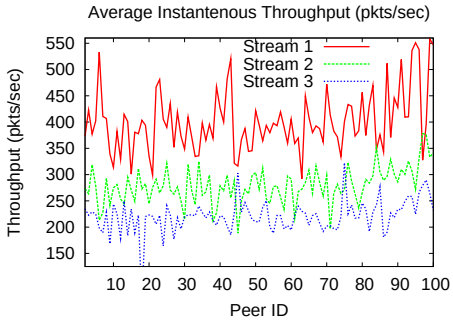


Fig. 11. Avg Instantaneous Throughput Per Peer for Streams 1, 2 and 3

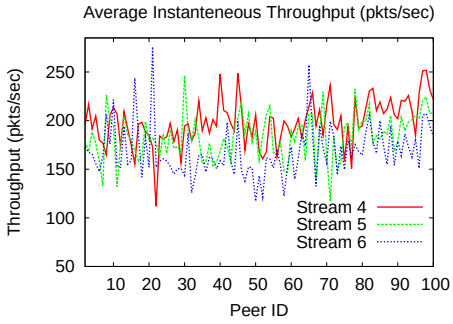


Fig. 12. Avg Instantaneous Peer Throughput for Streams 4, 5 and 6

show how efficiently ECDP enforces the priority based rate allocations.

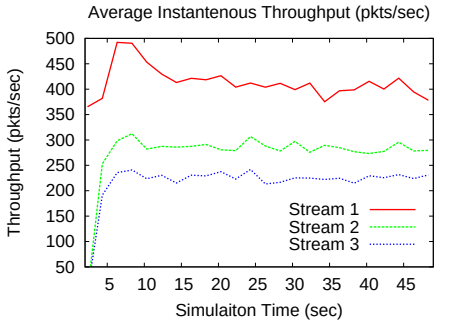


Fig. 13. Avg Instantaneous Throughput Over Time for Streams 1, 2 and 3

F. More Trace-Based Experiments

We have also conducted experiments based on the trace results presented in [26] for the content size distribution, [27] for the content popularity distribution and [28] for distribution of the flow arrival process. Since we could not obtain the raw trace data, we constructed the trace values (data points) from the plots given in these papers. We next present the trace extraction methodologies we used.

1) Extracting file size distribution: Based on the nature of the file size trace plot of US-Campus given in figure 4 of [26], we constructed piece-wise linear functions given by equation 15.

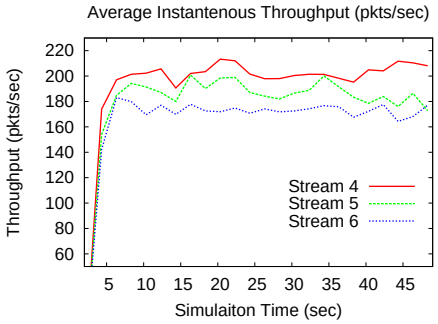


Fig. 14. Avg Instantaneous Throughput Over Time for Streams 4, 5 and 6

$$FS = \begin{cases} u(1,5), & cdf <= 0.17, \\ \frac{10-5}{0.18-0.17}(u(0,1)-0.17)+5.0, & 0.17 < cdf \leq 0.18, \\ \frac{200-10}{0.204-0.18}(u(0,1)-0.18)+10, & 0.18 < cdf \leq 0.204, \\ \frac{1000-200}{0.25-0.204}(u(0,1)-0.204)+200, & 0.204 < cdf \leq 0.25, \\ \frac{30000-1000}{0.96-0.25}(u(0,1)-0.25)+1000, & 0.25 < cdf \leq 0.96, \\ \frac{70000-30000}{0.99-0.96}(u(0,1)-0.96)+30000, & 0.96 < cdf \leq 0.99, \\ \frac{100000-70000}{1.0-0.99}(u(0,1)-0.99)+70000, & 0.99 < cdf \leq 1.0. \end{cases} \quad (15)$$

In equation 15, the function $u(a,b)$ generates a uniform random number between a and b and cdf is the CDF of the file size trace plot.

2) Extracting content popularity distribution: A Gamma distribution curve with a shape parameter of $\tilde{k} = 0.372$ and a scale parameter of $\theta = 23910$ is fitted to Youtube video content popularity distribution traces in figure 7 of [27]. The content popularity distribution in the paper which refers to the number of views of videos considers about $N_V = 1.6 \times 10^5$ videos. We normalized the scale parameter θ of the distribution by the number N_V of distinct videos so as to use it with simulation studies involving a different number of videos. The normalization steps are as follows.

With n_v the total number of video flows to be simulated, and p_v the average popularity of the videos, $n_v p_v$ is the total number of videos to be simulated. With a simulation time of t_s seconds and average video request arrival rate of λ_s flows per second, we have

$$n_v = \frac{t_s \lambda_s}{p_v}. \quad (16)$$

To obtain p_v, we normalize the number N_V of traced videos by the mean $\tilde{k}\theta$ of the Gamma popularity distribution as

$$\frac{n_v}{p_v} = \frac{N_V}{\tilde{k}\theta}. \quad (17)$$

Combining equations 16 and 17, we get the popularity value as

$$p_v = \sqrt{\frac{\tilde{k}\theta t_s \lambda_s}{N_V}}. \quad (18)$$

Using equation 18 in equation 16 we also obtain the number of distinct videos in the simulation.

3) Flow arrival distribution: We used the distribution of the number of flow arrivals per second given in [28] for our simulation. The paper fits a Poisson distributed curve to the trace and hence we used such a Poisson distribution for our flow arrival distribution. The number of YouTube servers (servers with unique IP addresses) used in the experiment was 2138. To scale our simulation we considered arrival rates to 1 and 10 servers. The experiment can simply be run for all servers with powerful machines.

4) More Trace Experimental Results: To compare the performance of pure ECDP based approach against other TCP based approaches, we considered the best case scenario for the TCP based approaches. This scenario uses the ECDP content selection mechanism. So using this same server selection mechanism we compared the performance of the TCP-based approaches (PACE, Dandelion) with our pure ECDP based approach. As can be seen from figures 15, 16 and 17, the pure ECDP approach gives lower file completion time when compared with TCP-based ECDP approach. For all experiments in this section, each YouTube file is divided into 50 chunks. So bigger file sizes have bigger chunk sizes. The YouTube video files we consider in this analysis are not live videos. Hence we use a content age of 15.5 seconds. This implies that videos which were first requested less than 15.5 seconds ago can still be requested. For all experiments of one YouTube server, the machine we used allowed us to run the simulation for 120 seconds. For the 10 YouTube servers experiments, we used a simulation time of 30 seconds.

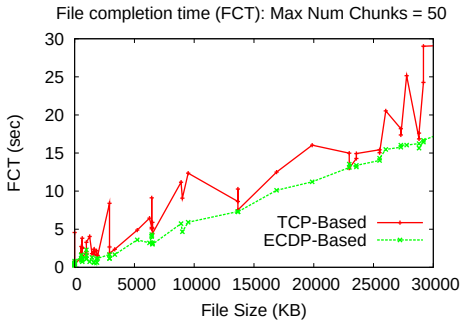


Fig. 15. File completion time with 1 YouTube server

Figure 16 shows the average file completion time (AFCT) of files less than 4000KB in size while figure 15 shows FCT of all files. As can be seen from figure 17, with more YouTube servers, the number of simulated peers requesting for content increases. This inturn increases the number of peers with a content and hence decreasing the file download time (AFCT). This is one of the noble gains of peer to peer systems as more peers means more bandwidth.

Figures 17, 20 and 19 show that overwhelming majority of the peers do not have to spend money to download GB of data as the credit amount they earn balances out with the amount they pay. For each peer, the amount to spend in these plots is calculated as the total amount of money a peer earns minus the total amount a peer has to pay per GB of content.

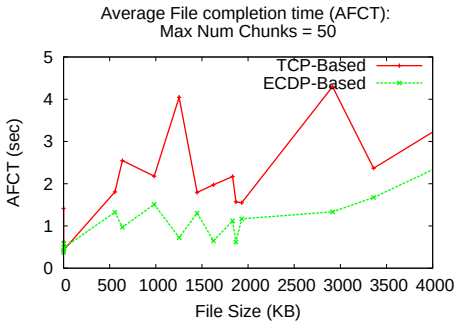


Fig. 16. Average file completion time (AFCT) with 1 YouTube server (small files)

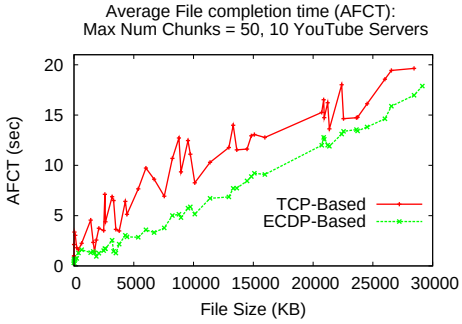


Fig. 17. Average file completion time (AFCT) with 10 YouTube server

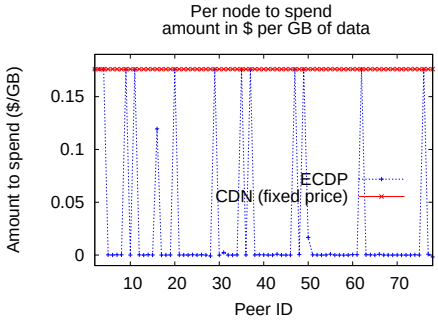


Fig. 18. Net amount to pay in dollars per GB of downloaded content with 1 YouTube server

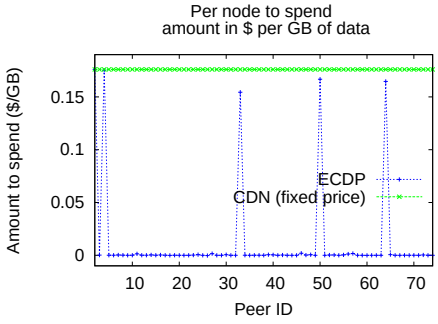


Fig. 19. Net amount to pay in dollars per GB of downloaded content with 10 YouTube servers (First few peers)

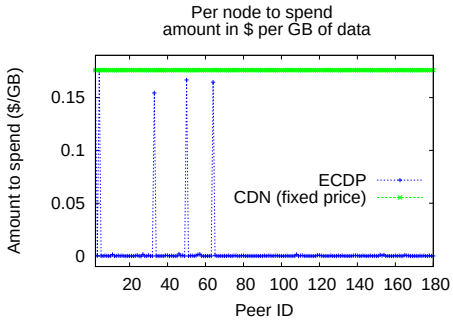


Fig. 20. Net amount to pay in dollars per GB of downloaded content with 10 YouTube servers (All peers)

Comparing figures 17 and 20, it can be seen that more YouTube servers in the experiment means more participating peers. The more peers have the contents the less other peers have to download the content from the CDN servers. This saves peers more money as can be seen from the plots. In all cases, the amount peers pay for bandwidth to download a content is less than the fixed CDN bandwidth amount charged by AmazonCLoudFront. For the experiments with only one YouTube server, the simulation generates fewer peers to download the content. As the number of peers which have the content is smaller, more peers download contents from the CDN servers paying more money as can be seen in figure 17. The amount which peers pay to directly download a content from the CDN servers can be subsidized (paid for) by the content providers as such peers are serving as seeders for the content provider.

G. CID Implementation Experiments

We have also implemented the basic features of ECDP in an Apache SQL server using PHP script. We implemented all the tables of the CID in an Ubuntu virtual machine using a quad four processor and a 1GB RAM. We generated *tblSelected-Source* table using a SELECT query from the tables *tblPeer*, *tblContent*, *tblPeerContent* as discussed in section V-A4 above. The tables are linked in a many-to-many relationship.

To see the performance gain of using the *tblSelectedSource* table over generating the contents requested by peers on the fly from the three tables, we have conducted experiments using and not using the *tblSelectedSource* table. We used one million records in each table for this experiment. As can be seen from figures 21 and 22 preparing the *tblSelectedSource* table as its source tables are updated results in significant gain in query time. Here, query time is the time from when a query for a specific record is made to when the reply is displayed from the SQL server. In these experiments we first generated uniform random content index records with the given contentIDs to request from the SQL server. The content with the ID of *cont396224* was the first content requested. Such initial request of a record resulted in a higher query time perhaps because the SQL server took time to upload parts of the table into memory. Figure 22 also shows that the query time increases with the increase in the record ID. This is because the tables are roughly sorted by requestIDs as the none-numeric parts of the contentID and peerID values are the same while both fields have text data types.

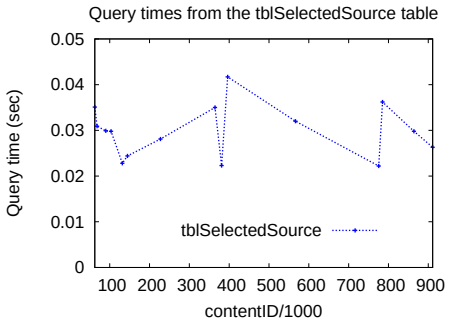


Fig. 21. Query time using the tblSelectedSource table

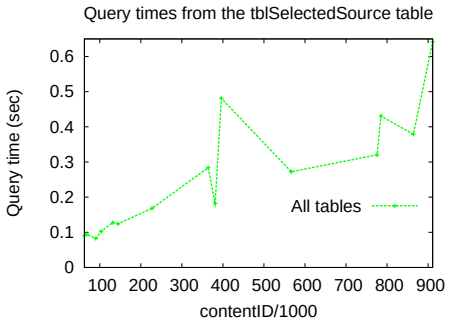


Fig. 22. Query time using SQL JOIN from all tables

The SQL query we made from the *tblSelectedSource* for the content with contentID of *cont396224* is as follows.

```
SELECT *
FROM 'tblSelectedSource'
WHERE contentID = 'cont396224'
LIMIT 0 , 1
```

And the following is the query we made from the three tables.

```
SELECT tblPeerContent.contentID, tblPeerContent.peerID,
  tblPeerContent.contentUrl, tblPeerContent.contentKey,
  tblContent.contentDesc, tblContent.contentPopularity,
  tblPeer.peerURate, tblPeer.peerUPrice, tblPeer.ratePerPrice
FROM tblPeerContent
INNER JOIN tblPeer ON tblPeerContent.peerID = tblPeer.peerID
INNER JOIN tblContent
  ON tblPeerContent.contentID = tblContent.contentID
WHERE tblPeerContent.contentID = 'cont396224'
LIMIT 0 , 1
```

We next conducted an experiment to know how long it takes for a query such as requesting the *contentKey* by a peer from the CID of the CIM. The propagation delay from the requesting peer virtual machine to the virtual machine with the SQL server is about 1ms. The times it takes for such query is shown in figure 23. There is a spike on the record of *cont132913* which is the first record requested by the peer in the experiment. Such a spike disappears with the other

requested records as perhaps the SQL server caches the session and keeps the *tblRequestedContent* table loaded in memory.

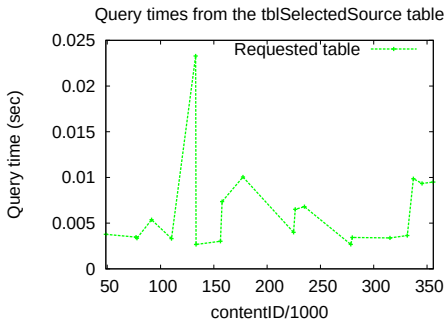


Fig. 23. Query time from peers to the CIM

The query we used for the experiments in figure 23 is as follows.

```
SELECT * FROM tblRequestedContent
WHERE contentID = 'cont$value'
LIMIT 0,1
```

These above query time figures are intended to show that the time it takes to resolve a certain query is not high even using a computer (server) with very limited hardward such as an Ubuntu virtual machine.

XII. Related Work

Over time, Peer-to-peer (P2P) content distribution has evolved to incorprate incentives in order to prevent freeloading. The BitTorrent [11], [12] uses a rate based tit-for-tat mechanism where users can achive higher download rate from peers to which they are uploading. In this case a peer which is not downloading a content is not incentivized to upload a content. In ECDP all peers are incentivized to continue uploading as every upload increases their credit maintained by the ECDP CIM. Reputation based schemes such as [29] help peers find another peer with the highest reputation score to download content from. Such a reputaiton scheme does not provide an accurate evaluation mechanism to choose a peer to serve a content. For instance a peer which is uploading many files without downloading a file can have a high reputation score. If such a peer does not have as much available upload capacity as another peer which is downloading files, peers will select it anyways becuase it has a high reputation score.

In the KARMA [30] scheme every peer has a set of managers which form banks which coordinate credit transfer with other peers. In this scheme there is no guarantee of integrity of the global currency when the majority of the managers are malicious. In ECDP a central CIM which cannot be manipulated by peers offers real monetary rewards to all peers which upload contents. PACE [14] uses bandwidth pricing to help uploading peers earn credit. However PACE does not give a fair-exchange of content for payment as the content demand at a peer is estimated as a total requested download rate at remote buy clients. Such demand used to obtain a bandwidth price is not peer specific. Dandelion [8] is based on a centralized online currency bank mechanism to incentivize peers. However Dandelion uses a fixed pricing mechanism that peers are not awarded according to the upload bandwidth they offer to upload contents. Peers do not decrease their price to attract more customers when they have high upload rate and viceversa. PRIME [31] is a mesh-based P2P streaming. Eventhough it tries to balance the average outgoing rate of a source peer with the average incoming rate of a content receiving peer, it does not use an efficient rate allocation and enforcement mechanism like ECDP to achieve a max/min allocation. It uses a TCP friendly rate control protocol (TFRC) [32] which inherits the TCP problems of not quickly utilizing avaialable link capacities. In PRIME each peer tries to maintain many parents that can collectively serve as content providers using a mesh-based overlay constraction which can potentially incure significant overhead. Unlike ECDP, PRIME does not give an efficient mechanisim to help peers select a content source with high throughput and minimum bandwidth cost. This is because a new peer selects a random subset of peers to be its content parents. A reliable client accounting system of a commercial hybrid content-distribution network (Akamai) is also presented in [33] to detect and mitigate a variety of attacks by malicious peers. This mechanisim improves the NetSession which is a peer-assisted content delivery network (CDN) operated by Akamai. In ECDP peers do have any incentive to act maliciously. This is because peers get monetary incentives (credit) for uploading content and all transactions are co-ordinated by a scalable centralized ECDP CIM. If an ECDP peer acts maliciously, it only wastes its bandwidth and suffers monetary losses.

A hybrid CDN-P2P system for live video streaming called LiveSky is presented in [7]. The paper gives a trace based study of extensive LiveSky deployment in China. However the work only gives approximate guideline for peer selection. For instance the paper assume that the total upload bandwidth of clients in level k of the P2P tree is always larger than the download bandwidth requirement of clients in level k+1. It also only considers aggregate measures (i.e., population and time averages) to model the end-user properties. On the other hand ECDP does not make such assumptions and uses accurate rate and price based incentives to select content sources to serve a content. This gives peers a reliable incentive to cooperate without a malice. LiveSky also limits peer selection to a local network while ECDP does not make that ristriction unless local content source selection stragy is used or the local peers have the best upload rate and lowest prices. A study in [34] shows that redirecting every client to the CDN server with least latency does not suffice to optimize client latencies. The authors of this paper proposed a system called *WhyHigh* to optimize Google CDN perfomance. WhyHigh measures client latencies across all nodes in the CDN and correlates measurements to identify the prefixes affected by inflated latencies. ECDP by design chooses peers or CDN servers which offer the highest throughput and lowest price and does not require complex in efficient systems such as WhyHigh to select content sources.

NetTube, a P2P assisted content delivering framework that explores the clustering in social networks for short video sharing is proposed in [35]. Like NetTube, ECDP allows users to share their contents while keeping it in their own servers. Unlike ECDP, NetTube selects a content source based on social groups and not based on throughput and bandwidth price. SocialTube, which is peer-assisted video sharing system that explores social relationship, interest similarity, and physical location between peers in online social networks (OSNs) is proposed in [36]. SocialTube uses a social network (SN)-based P2P overlay construction algorithm. Unlike ECDP, SocialTube does not select content sources based on a high upload bandwidth and low cost. This can result in SocialTube unnecessarily delaying streaming and other content transfer when other peers not in the same social group with high upload capacity exist.

Besides, unlike ECDP, all the above schemes do not help peers determine an accurate rate at which they can download content from other peers. They do not give a mechanism to prioritize content transfers which is an important component of 3D [20] and other streaming applications. Unlike ECDP they also do not provide an efficient max/min rate allocation mechanisim.

XIII. Conclusion

In this paper we proposed the design of an efficient content distribution protocol (ECDP). Unlike previous distributed content distribution attempts, ECDP relies on an accurate and fair incentive mechanism which allows prioritized rate allocations and enforcements. ECDP is a flexible scheme which allows multiple server selection strategies and can achieve max/min allocation. Unlike previous work we have presented a noble content index management scheme for ECDP.

We have implemented ECDP in the NS2 simualation package. We evaluated the performance of ECDP using rigurous trace based simulation experiments. The experiments demonstrate the ECDP design goals of allocations and enforcements. We have also implemented ECDP in Apache SQL server using PHP in Ubuntu virtual machines. The implementation experiments show that ECDP can easily scale to millions of content index records. We have also shown how ECDP can be deployed in the current Internet architecture with siginificant gains.

References

[1] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. –, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=2043164.1851194

[2] C. Labovitz, "Internet traffic evolution 2007 - 2011," http://www.monkey.org/ labovit/papers/gpf_2011.pdf.

[3] W. Wu, A. Arefin, Z. Huang, P. Agarwal, S. Shi, R. Rivas, and K. Nahrstedt, ""i'm the jedi!" - a case study of user experience in 3d tele-immersive gaming," in *Proceedings of the 2010 IEEE International Symposium on Multimedia*, ser. ISM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 220–227. [Online]. Available: http://dx.doi.org/10.1109/ISM.2010.39

[4] M. Marcon, B. Viswanath, M. Cha, and K. P. Gummadi, "Sharing social content from home: a measurement-driven feasibility study," in *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*, ser. NOSSDAV '11. New York, NY, USA: ACM, 2011, pp. 45–50. [Online]. Available: http://doi.acm.org/10.1145/1989240.1989253

[5] "The diaspora* project," http://diasporaproject.org/.

[6] K. Weise, "On diaspora's social network, you own your data," http://www.businessweek.com/articles/2012-05-10/on-diasporas-social-network-you-own-your-data, 2012.

[7] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM international conference on Multimedia*, ser. MM '09. New York, NY, USA: ACM, 2009, pp. 25–34. [Online]. Available: http://doi.acm.org/10.1145/1631272.1631279

[8] M. Sirivianos, X. Yang, and S. Jarecki, "Robust and efficient incentives for cooperative content distribution," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1766–1779, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1109/TNET.2009.2021655

[9] P. Wendell and M. J. Freedman, "Going viral: flash crowds in an open cdn," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 549–558. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068867

[10] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid cdn-p2p: why limelight needs its own red swoosh," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '08. New York, NY, USA: ACM, 2008, pp. 75–80. [Online]. Available: http://doi.acm.org/10.1145/1496046.1496064

[11] B. Cohen, "Incentives build robustness in bittorrent," Proc. Workshop Econ. Peer-to-Peer Syst, 2003.

[12] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bit torrent," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973431

[13] F. Wu and L. Zhang, "Proportional response dynamics leads to market equilibrium," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, ser. STOC '07. New York, NY, USA: ACM, 2007, pp. 354–363. [Online]. Available: http://doi.acm.org/10.1145/1250790.1250844

[14] C. Aperjis, R. Johari, and M. J. Freedman, "Bilateral and multilateral exchanges for peer-assisted content distribution," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1290–1303, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1109/TNET.2011.2114898

[15] C. Aperjis, M. J. Freedman, and R. Johari, "Peer-assisted content distribution with prices," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08. New York, NY, USA: ACM, 2008, pp. 17:1–17:12. [Online]. Available: http://doi.acm.org/10.1145/1544012.1544029

[16] M. Sirivianos, J. H. Park, X. Yang, and S. Jarecki, "Dandelion: cooperative content distribution with robust incentives," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, ser. ATC'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 12:1–12:14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364385.1364397

[17] S. McCanne and S. Floyd, "NS-2," http://www.isi.edu/nsnam/ns/.

[18] P. Pillay-Esnault., P. Moyer, J. Doyle, E. Ertekin, and M. Lundberg, "Ospfv3 as a provider edge to customer edge (pe-ce) routing protocol," United States, 2012.

[19] D. Fesehaye, P. Xia, P. B. Godfrey, K. Nahrstedt, and S. S. Lumetta, "NCP: Finishing Flows Even More Quickly," *University of Illinios, Department of Computer Science, Technical Report*, mar 2012. [Online]. Available: http://www.ideals.illinois.edu/handle/2142/30401

[20] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy, "Enabling multi-party 3d tele-immersive environments with viewcast," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 6, no. 2, pp. 7:1–7:30, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1671962.1671963

[21] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[22] Internet2, "Internet2 Network: GlobalNOC RealTimeAtlas," http://atlas.grnoc.iu.edu/I2.html, 2012.
[23] P.-A. Larson, C. Clinciu, E. N. Hanson, A. Oks, S. L. Price, S. Rangarajan, A. Surna, and Q. Zhou, "Sql server column store indexes," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, ser. SIGMOD '11. New York, NY, USA: ACM, 2011, pp. 1177–1184. [Online]. Available: http://doi.acm.org/10.1145/1989323.1989448
[24] "Planetlab 4hr traces," http://www.eecs.harvard.edu/syrah/nc/sim/pings.4hr.stamp.gz.
[25] Amazon, "Amazon cloudfront," http://aws.amazon.com/cloudfront, 2012.
[26] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting video server selection strategies in the youtube cdn," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ser. ICDCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 248–257. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2011.43
[27] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, june 2008, pp. 229 –238.
[28] T. Mori, R. Kawahara, H. Hasegawa, and S. Shimogawa, "Characterizing traffic flows originating from large-scale video sharing services," in *Proceedings of the Second international conference on Traffic Monitoring and Analysis*, ser. TMA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 17–31.
[29] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, ser. NOSSDAV '03. New York, NY, USA: ACM, 2003, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/776322.776346
[30] V. Vishnumurthy, S. Chandrakumar, , and E. G. Sirer., "Karma: A secure economic framework for peer-to-peer resource sharing," Proc. Workshop on Economics of Peer-to-Peer Systems, 2003.
[31] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 4, pp. 1052 –1065, aug. 2009.
[32] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "Tcp friendly rate control (tfrc): Protocol specification," United States, 2003.
[33] P. Aditya, M. Zhao, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, and B. Wishon, "Reliable client accounting for p2p-infrastructure hybrids," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 8–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228309
[34] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize cdn performance," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 190–201. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644917
[35] X. Cheng and J. Liu, "Nettube: Exploring social networks for peer-to-peer short video sharing," in *INFOCOM*, 2009, pp. 1152–1160.
[36] Z. Li, H. Shen, H. Wang, G. Liu, and J. Li, "Socialtube: P2p-assisted video sharing in online social networks," in *INFOCOM*, 2012, pp. 2886–2890.