# Decentralized P2P Protocol for Video-on-Demand Streaming: Simple and Efficient

Wenbin Tang, Xiaowei Wu
Department of Computer Science, the University of Hong Kong
{wbtang, xwwu}@cs.hku.hk

***Abstract*—We consider the video-on-demand streaming problem in a P2P network in a decentralized model, in which peers have no global information about the network. Assuming that only one *server* has all the chunks, the objective is to stream all chunks to all peers in the network such that small *latency* and good *fluency* are achieved by all peers.**

**We design a simple and decentralized protocol in which each peer maintains a constant number of neighbors and only need to communicate with one of them chosen uniformly at random every time. Moreover, the maximum number of communications established on each peer every time is also constant. We provide theoretical and experimental analysis to show that almost all peers achieve optimal latency and fluency under our protocol.**

## I. Introduction

The Peer-to-Peer (P2P) network has become a major player on Internet in recent years. The efficiency of P2P assisted downloading has already been proved by many popular P2P file downloading systems like BitTorrent and Emule. Recent attention is drawn by the P2P video-on-demand (VoD) in which peers may be interested in playing different portions of the same video [1], [2]. Peers in those systems contribute their own upload bandwidths so that a large population of peers can be served at the same time. To achieve perfect pipelining, the video is divided into many small fractions, which are usually called *chunks*. Each chunk can be transmitted in a short period of time, which we denote by a *round*.

To overcome the difficulty of balancing the downloading requests among all peers, it is implicitly assumed by past researchers that there are many peers that have cached the entire video (all chunks) and there exists a central entity that controls and helps peers to make efficient communications [3], [4], [5]. It is hence challenging and interesting to consider *decentralized* systems in which peers have very limit view of the whole network [6], [7], [8]. We consider in this paper a highly decentralized model: (1) the tracker only knows the identities of all peers currently in the network, and nothing else; (2) each peer is randomly assigned a constant number of peers as neighbors [9]; (3) each peer only knows its own set of neighbors and makes communication with one of them randomly chosen every round.

In contrast to live-streaming, under the VoD setting, peers may start watching the video at different positions [1], [2]. Based on experimental results [1], it is assumed that about half of the peers perform random-seek and the starting position (chunk) follows some exponential distribution. In general, peers will leave and join the system frequently, especially when some peers have finished watching the whole video. Based on the observation of steady networks, it is assumed that the number of peers in the network is fixed [7], [8]. We also adopt this assumption in our model.

### A. Related Work

Huang et al. [1] analyzed the P2P VoD streaming in a chunk-based model, in which peers can choose their starting positions randomly and may jump back and forth. Assuming the existence of a set of servers and trackers, they carried out a large scale experimental analysis to quantify the user behaviors, the effectiveness and the level of user satisfaction. Compared to their settings, we assume in our model that there is only one server and the streaming is carried out in a fully decentralized setting. Similar model was also adopted by Parvez et al. [10]. Our model is different from theirs in two aspects: 1) in our model, peers not only help their neighbors on buffering chunks, but also on spreading requests of chunks; 2) we measure our protocol by the latency and fluency of each peer in the network, which are not quantified in their model.

A theoretical analysis is provided by Zhao et al. [8] for P2P VoD under the decentralized assumption. Each peer is assigned $\Theta(\log N)$ number of neighbors, where $N$ is the number of peers in the network. Their flow-based analysis mainly focused on achieving an optimal streaming rate for all peers, while the details on how the chunks are selected and transmitted are not mentioned. Our protocol improves their result by restricting the number of neighbors of each peer and the connections each peer can make during each round to constant.

Recently, Tang et al. [11] analyzed the P2P VoD problem and proposed a similar chunk-based protocol. They showed by experiments that their protocol achieves small latency and good fluency for most of the peers. However, no theoretical analysis is provided for their protocol and the comparison provided is inadequate. We improve their result by providing a simplified protocol, which is more efficient, together with theoretical analysis on the performance.

### B. Our Contribution

In this paper, we propose a simple protocol in which the peers communicate with each other under fully decentralized control. We provide theoretical analysis to show upper and lower bounds for our protocol performance. Our proof also provides insights into the efficiency of the protocol, which are later verified by our simulation. We design detailed simulation for the model and our protocol. Experimental result suggests that peers achieve almost-optimal latency ($\approx 0$) and fluency ($\approx 1$) under our protocol. Compared to past literatures, our contributions can be summarized as follows.

The model we consider is more general. The network in which the peers are connected is sparse (constant in-degree) and dynamic. Only one peer in the network has all chunks. Moreover, only a constant number of connections can be established on each peer every round, including the server.

Our theoretical analysis guarantees that the latency of each peer in our protocol is at most $O(\log^2 N)$, which is an $O(\log N)$ approximation of any optimal protocol.Moreover, we provide non-trivial upper and lower bounds related to the communication details, which give key insights as to why our simple protocol achieves almost-optimal performance.

## II. Video Streaming Model

Table I lists some important notations and their meanings that will be used throughout this paper.

TABLE I: Summary of Notations

| **Notations** | **Meaning** |
|---|---|
| $N$ | The number of peers in the network |
| $M$ | The number of chunks to be transmitted |
| $C$ | The number of neighbors of each peer |
| $K$ | Maximum number of accepted connections for each peer |
| $\epsilon$ | Fraction of peers that perform random-seek |
| $C_i$ | The $i^{th}$ chunk |
| $P_i$ | The $i^{th}$ peer |
| $S_i$ | The start position of the $i^{th}$ peer |
| $l_i$ | The latency of the $i^{th}$ peer |
| $f_i$ | The fluency of the $i^{th}$ peer |
| $R$ | Maximum number of requests transmitted per communication |

We consider the video streaming problem in a network consist of $N$ peers $\{P_1, P_2, \ldots, P_N\}$. Assuming that at the beginning, only the *server* $P_1$ owns the video, which is divided into $M$ consecutive *chunks* $\{C_1, C_2, \ldots, C_M\}$, the problem is to stream all the chunks to all peers in the network. To obtain fluency, each peer will try to get the chunks after its current viewing position as soon as possible.

To simplify the analysis, we discretize the continuous time into *rounds* [12]. All communications between peers will be made in a round-by-round manner. To simulate the video streaming in a real network as detailed as possible, the following settings are made.

*1) Bounded Bandwidth:* Although in real world, each peer may have different bandwidth limit [13], to ensure that all peers achieve perfect streaming, in our model each peer (including the server) only utilizes a constant number $K$ of communications with its neighbors.

*2) Decentralized Communication:* We assume in our model that all communications between peers are made in a *decentralized* manner. Each peer in the network only asks for chunks from its own neighbors and has no global information about the network. The selection of communication partners and chunks are all made *locally* and randomly.

*3) Video-on-Demand:* It has been a common assumption in recent years that a constant $\epsilon$ fraction of peers will start watching the video from random positions [1], which is called *random-seek*. Let $S_i \in [M]$ be the starting position of peer $P_i$. We assume that each $S_i$ is identically and independently distributed as : $Pr[S_i \geq j] = e^{-\frac{3j}{M}}, \forall j \in [M]$. The same assumption is also made in [1], [8], [11].

*4) Dynamic Network:* Peers may leave the network during downloading (with probability (prob.) $\Pr(\text{exit})$) or after the watching (with prob. 1). To simplify the setting, we assume that the number of peers in the network is always $N$.

### A. Measurements

The protocol for the video streaming problem can be evaluated by the *latency* and *fluency* of each peer in the network. We call a peer $P_i$ *waiting* if it has not received $C_{S_i}$ (the chunk at its start position) and *downloading* otherwise. If peer $P_i$ has collected all chunks after position $S_i$, then we call it *finished*.

Let $t_i^{(1)}$ be the round when $P_i$ joins the network; $t_i^{(2)}$ be the round when $C_{S_i}$ is received; $t_i^{(3)}$ be the round when $P_i$ leaves the network. For each peer $P_i$ that is downloading, we call a round *bad* if $P_i$ has consumed all chunks it has downloaded and is trying to download the next chunk and *good* otherwise.

**Definition 1** (Latency). *The latency $l_i$ for $P_i$ is the number of rounds before $P_i$ receives chunk $C_{S_i}$, which is $l_i = t_i^{(2)} - t_i^{(1)}$*

**Definition 2** (Fluency). *The fluency $f_i$ for $P_i$ is defined to be the ratio between the number of good rounds $g_i$ and the total number of rounds after $C_{S_i}$ is received, which is $f_i = \frac{g_i}{t_i^{(3)} - t_i^{(2)}}$.*

In application, short latency implies that a peer can start watching the video soon after it joins the network. Note that $f_i \in (0, 1]$ for each peer $P_i$ and larger $f_i$ implies a better fluency since less time is spent on buffering. Based on the above definition, we provide a lower bound for the average latency under the problem setting, which is a generalized version of the result obtained by Yang and Veciana [14].

**Theorem 1.** *Assume that there are $N$ waiting peers in the Network. Under the limit of bandwidth (at most $K$ accepted connections every round), there does not exist any protocol that achieves average latency $o(\log_K N)$.*

*Proof.* Notice that only the connection between a downloading peer and a waiting peer can potentially increase the number of downloading peers. Let $N_t$ be the number of downloading peers at the end of round-$t$. Then we have $N_0 = 1$.

Since each peer can establish at most $K + 1$ connections, we have $N_{t+1} \leq (K + 1)N_t$ for all $t \geq 0$. Hence it is easy to observe that the number of rounds before $\frac{N}{K+1}$ becomes downloading is at least $\log_{K+1} \frac{N}{K+1} = \Theta(\log_K N)$. $\square$

Theorem 1 demonstrates that it is impossible to achieve a constant latency when there are $N$ waiting peers in the network at the same time. The lower bound provided in the theorem will be used later to show that our protocol obtains a non-trivial approximation ratio compared to any other protocol.

## B. Network Topology

There are $N$ peers connected by a BTC network $G_N$. Each peer $P_i$ is assigned a constant number of $\min\{i-1, C\}$ peers chosen uniformly at random from all existing peers $\{P_1, P_2, \ldots, P_{i-1}\}$ as neighbors, for all $i \geq 2$.

**Definition 3** (BTC-Network)**.** *Stating from 1 source peer $P_1$, peer $P_i$ ($2 \leq i \leq N$) is added to the network and connected to $\min\{i-1, C\}$ peers chosen uniformly at random (without replacement) from all existing peers $\{P_1, P_2, \ldots, P_{i-1}\}$.*

Next we analyze the diameter of $G_N$ and the degrees of peers. We also show that these network properties hold even when the network is dynamic. The following lemmas are important for the performance analysis of our protocol.

**Lemma 1.** *Diameter of $G_N$ is $O(\log_C N)$ with prob. $1-o(\frac{1}{N})$.*

*Proof.* First we observe that each peer $P_i$ will choose $\min\{i-1, C\}$ peers from $\{P_1, P_2, \ldots, P_{i-1}\}$ as neighbors and hence the graph $G_N$ must be connected. Let dist$(i)$ be the shortest distance between $P_i$ and $P_1$ in $G_N$.

Let $\hat{P}(i) = \min\{t | P_t \text{ is a neighbor of } P_i\}$ be the neighbor of $P_i$, where $i \geq 2C$, with smallest index. Note that $\hat{P}(i) \in [i-1]$ and $E[\hat{P}(i)] = \frac{i}{C}$. By Markov's inequality, we have $\Pr\left[\hat{P}(i) \leq \frac{2i}{C}\right] = 1 - \Pr\left[\hat{P}(i) \geq \frac{2i}{C}\right] \geq \frac{1}{2}$. We call peer $P_i$ *good* if $\hat{P}(i) \leq \frac{2i}{C}$. Hence for each $i \geq 2C$, peer $P_i$ is good with prob. at least $\frac{1}{2}$ and is independent of all other peers.

Let $\hat{P}^{(1)}(i) = \hat{P}(i)$ and $\hat{P}^{(x)}(i) = \hat{P}(\hat{P}^{(x-1)}(i))$ be the peer obtained by applying $\hat{P}(*)$ for $x$ times starting from $P_i$. Note that if $\hat{P}^{(x)}(i) \leq C$, then we have dist$(i) \leq x$.

Observe that if the number of good peers along the path $\{i, \hat{P}^{(1)}(i), \hat{P}^{(2)}(i), \ldots, \hat{P}^{(x)}(i)\}$ is at least $\log_{\frac{C}{2}} N$, then $\hat{P}^{(x)}(i) \leq C$. Let $\tilde{P_i^x} = \{i, \hat{P}^{(1)}(i), \hat{P}^{(2)}(i), \ldots, \hat{P}^{(x)}(i)\}$ be the path of length $x+1$. Recall that each peer in $\tilde{P_i^x}$ is good with prob. at least $\frac{1}{2}$, independently.

Hence by Chernoff Bound we have $\Pr[\text{dist}(i) > 8C\log_{\frac{C}{2}} N] \leq \Pr[\hat{P}^{(8C\log_{\frac{C}{2}} N)}(i) > C] \leq \Pr[\text{number of good peers} \in \tilde{P_i}^{8C\log_{\frac{C}{2}} N} < \log_{\frac{C}{2}} N] \leq \exp(-\frac{1}{2} \times \left(\frac{8C-1}{8C}\right)^2 \times 4C\log_{\frac{C}{2}} N) = o(\frac{1}{N^2})$. By taking union bound over all peers $P_i$ with $i \geq 8C\log_{\frac{C}{2}} N$, we conclude that with prob. $1-o(\frac{1}{N})$, the diameter of $G_N$ is at most $8C\log_{\frac{C}{2}} N = O(\log_C N)$. □

The following two lemmas analyze the expected degree of each node and the measure concentration.

**Lemma 2.** *Let deg$(i)$ be the degree (including out-edges and in-edges) of peer $P_i$ in $G_N$. Then for all $i > C$ we have $E[\text{deg}(i)] = C(1 + \sum_{j=i}^{N-1} \frac{1}{j}) = \Theta(C \log \frac{N}{i})$.*

*Proof.* For each $i \geq C+1$, we know that deg$(i) = C$ when $P_i$ joins the network; for all $j > i$, $P_j$ is connected to $P_i$ with prob. $\frac{C}{j-1}$. Hence we have $E[\text{deg}(i)] = C + \sum_{j=i+1}^{N} \frac{C}{j-1} = C(\sum_{j=1}^{N-1} \frac{1}{j} - \sum_{j=2}^{i-1} \frac{1}{j}) = \Theta(\log \frac{N}{i})$. □

By Lemma 2 and measure concentration results (i.e., Chernoff Bound), we obtain the following lemma immediately.

**Lemma 3.** *With prob. $1 - o(\frac{1}{N})$, the number of peers with constant degree in $G_N$ is $\Theta(N)$; for each $i = o(N)$, we have deg$(i) = \Theta(\log \frac{N}{i})$ with prob. $1 - o(1)$.*

Lemma 2 indicates that most of peers in $G_N$ are of constant degree. Moreover, the following dynamic setting guarantees that all the above structural properties of $G_N$ will hold even when $G_N$ is dynamic. Once a peer $P_i$ leaves the network, a new peer will join and make connections to $C$ neighbors chosen uniformly at random from all $N-1$ existing peers. For each $P_j$, $j > i$, that is a neighbor of $P_i$ (before $P_i$ leaves), a new peer that is not neighbor of $P_j$ will be chosen uniformly at random from $\{P_1, \ldots, P_{j-1}\} \backslash \{P_i\}$ (if any) as the new neighbor of $P_j$.

**Lemma 4.** *The structural properties of $G_N$ as shown in Lemma 1, 2, 3 hold after the dynamic reconstruction.*

*Proof.* By the above restructuring, the network remains the same capacity of $N$ peers. More importantly, if we fix any round and relabel all peers in the network as $P_1, P_2, \ldots, P_N$ in the order of peers sorted by the time they joined the network (earliest to latest), then for all $i \geq C+1$, the $C$ neighbors of peer $P_i$ are chosen uniformly at random from $\{P_1, P_2, \ldots, P_{i-1}\}$. Hence at any round when the network is still running, all network topology properties (except for a constant number of peers) proved above hold with the same probabilities. □

## III. Prioritized Request Queue

In this section, we introduce the *Prioritized Request Queue* (PRQ) protocol to disseminate requests and chunks.

The set of chunks each peer $P_i$ has can be indicated by a length $M$ bit-set (*chunk bit-set*) $\{\beta_1, \beta_2, \ldots, \beta_M\}$ where $\beta_i = 1$ if $P_i$ has chunk $C_i$ and 0 otherwise. During the streaming procedure, each peer will choose one of its neighbors uniformly at random every round and make communication. The communication is divided into the following 3 steps.

*1) Connection (Algo. 1):* In the first step, the initiating peer sends the connection request to its neighbor. Among all connection requests, at most $K$ will be chosen for acceptance.

**Algorithm 1** PRQ Protocol: Step 1

```
1: procedure TryConnectANeighbor($P_i$)
2:     $P_i$.Connect($P_i.outNeighbors$.RandomPick(1))
3: procedure ProcessConnectionRequest($P_i$, $K$)
4:     for $P_j \in P_i.waitingList$.RandomPick(K) do
5:         $P_i$.Accept($P_j$)            ▷ accept at most K connections
6: procedure Connected(from: $P_i$, to: $P_j$)
7:     ExchangeRequest($P_i$, $P_j$)
8:     ExchangeChunk($P_i$, $P_j$)
9:     $P_i$.UpdatePRQ
```

A set of requests and at most 2 chunks will be transmitted for each established communication in the second step. Each request has the following 3 attached fields.

- ID. The index of requested chunk: ID $\in [M]$.
- Time-Stamp (TS). The Time-Stamp of a request is set to be the time by which the requested chunk should be

received so that the streaming can be carried out fluently. Time-Stamp indicates how "urgent" the chunk is required.

- Time-To-Live (TTL). The maximum number of rounds a request is allowed to live before it expires. We set TTL to $h$ when the request is generated and then decrease it by 1 every round.

Note that since the bandwidth of each peer is limited, TS helps us to identify those "urgent" requests and hence can prevent the "race condition" [15], which would hurt the network performance, from happening. Requests with TTL$=0$ will be deleted. We will later show that TTL plays an important role in controlling the number of copies of one request (Lemma 7).

*2) Exchange Data (Algo. 2):* In the second step, only the top-$R$ requests from the sorted requests list will be chosen for transmission, where $R$ is a constant. When the $R$ requests are received, the peer will try to send back the chunk it owns that is required by a request with a priority as high as possible.

**Algorithm 2** PRQ Protocol: Step 2

```
1: procedure SENDREQUEST(from: $P_i$, to: $P_j$, $R$)
2:     $P_i$.SENDTO($P_j$, $P_i.Req$.top($R$))                ▷ send R requests
3: procedure SENDCHUNK(from: $P_i$, to: $P_j$, $sent \leftarrow false$)
4:     for $r \in P_i$.REQUESTFROM($P_j$) do
5:         if $r.ID \notin P_i.chunks$ then
6:             $P_i.Req$.ADD(r)                          ▷ add requests to PRQ
7:         if !$sent$ and $chunk \leftarrow P_i.chunks$.FIND(r.ID) then
8:             $P_i$.SENDCHUNKTO($P_j$, $chunk$)
9:             $sent = true$                             ▷ send at most one chunk
```

*3) Update Requests (Algo. 3):* In the third step, the chunk bit-set will be updated to indicate the acquisition of new chunks. To help disseminating requests from neighbors, the list of requests will be updated as follows.

- The TTL of all requests are decreased by 1.
- Each peer that is not finished will generate a new request for the chunk after its current downloading position.
- Delete the requests with TTL$=0$ and requests asking for chunks that is already received.
- Merge the requests with the same ID into one: set its TTL to the highest and TS to the lowest.

Requests are sorted according to the following rules: (1) smaller TS first; (2) smaller TTL first; (3) smaller ID first.

**Algorithm 3** PRQ Protocol: Step 3

```
 1: procedure UPDATEPRQ($P_i$, $Req$, $L \leftarrow$ List) ▷ queued requests
 2:     if !$P_i.finished$ then                 ▷ generate its own request
 3:         $L$.ADD($P_i.NewRequest$)
 4:     for $r \in P_i.Req$ do                  ▷ $P_i.Req$ - buffered requests
 5:         r.TTL -= 1
 6:         if $r.TTL > 0$ and $r.ID \notin P_i.chunks$ then
 7:             if $t \leftarrow L$.FIND(r.ID) then           ▷ merge same ID
 8:                 $t.TTL$ = MAX(t.TTL, r.TTL)
 9:                 $t.TS$ = MIN(t.TS, r.TS)
10:             else
11:                 $L$.ADD($r$)
12:     $P_i.Req = L$.PRIORIZEDBY(small TS, small TTL, small ID)
```

The next theorem provides an upper bound for the latency of our protocol (assume that $K$ and $h$ are constant).

**Theorem 2.** *With prob. $1-o(1)$, latency of each peer in our protocol is $O(\log^2 N)$.*

*Proof.* By lemma 2, we have $E[\deg(i)] = \Theta(C\log\frac{N}{i})$. Let $\delta = \frac{4C^2\log N}{E[\deg(i)]} - 1$, we have $\frac{\delta^2}{2+\delta} \geq \frac{1+\delta}{2}$. By Chernoff Bound we have $\Pr[\deg(i) > 4C^2\log N] = \Pr[\deg(i) > (1+\delta)E[\deg(i)]] \leq \exp(-\frac{\delta^2}{2+\delta}E[\deg(i)]) \leq \exp(-\frac{1+\delta}{2}E[\deg(i)]) = \exp(-2C^2\log N) = o(\frac{1}{N^2})$. Taking union bound over all peers, then with prob. $1-o(\frac{1}{N})$, the maximum degree of $G_N$ is at most $4C^2\log N = O(\log N)$. If a connection request is sent to a peer of degree $O(\log N)$, then the request will be accepted with prob. $\Omega(\frac{1}{\log N})$.

By Lemma 1 we know that with prob. $1-o(\frac{1}{N})$, each peer $P_i$ is connected to the server $P_1$ by a path of length $O(\log N)$. Since each peer along the path between $P_i$ and $P_1$ has constant out-degree $C$, a connection request will be sent between any two consecutive peers with prob. $\frac{1}{C}$ and accepted with prob. at least $\Omega(\frac{1}{\log N})$. Moreover, by Lemma 7, we know that each request will be sent with a constant prob. via each established connection. By Fact 4.2 of [16], with prob. $1-o(1)$, the request for $C_{S_i}$ will be sent to $P_1$ in $O(\log^2 N)$ rounds and the chunk $C_{S_i}$ will be sent to $P_i$ in $O(\log^2 N)$ rounds. ☐

Theorem 1 implies that our protocol is an $O(\log N)$ approximation of any optimal protocol in terms of latency.

**Corollary 1.** *The PRQ protocol achieves an average latency that is an $O(\log N)$ approximation of the latency of any optimal protocol.*

Insights into the efficiency of our protocol are provided by the next few lemmas. We prove that in each round, the number of connection requests received by each peer, the fraction of idle peers in the whole network and the length of the PRQ of each peer are all bounded as follows.

**Lemma 5.** *The number of connection requests $P_i$ receives each round follows a binomial distribution $B(deg(i)-C, \frac{1}{C})$, which has expectation $\frac{E[deg(i)-C]}{C} \approx \ln\frac{N}{i}$.*

*Proof.* Note that $P_i$ has $\deg(i)-C$ out-neighbors and each of them has $C$ neighbors, from which one of them will chosen uniformly at random each round for communication, independently for each out-neighbor of $P_i$. Hence the number of connection requests $P_i$ will receive each round follows the binomial distribution $B(\deg(i)-C, \frac{1}{C})$.

By Lemma 2 we know that $E[\deg(i)-C] = C(\sum_{j=1}^{N-1}\frac{1}{j} - \sum_{j=1}^{i-1}\frac{1}{j}) \approx C\ln\frac{N}{i}$, which means the expected number of connection requests $P_i$ receives each round is $\frac{E[\deg(i)-C]}{C} \approx \ln\frac{N}{i}$ (for $i \ll N$). ☐

We call a peer $P_i$ *idle* in a round if there is no connection built between $P_i$ and any other peer. The fraction of idle peers is an indicator of network efficiency since no chunk will be transmitted from or towards an idle peer, which can be regarded as a loss of network efficiency.

**Lemma 6.** *The expected fraction of idle peers in each round is at most $e^{-\frac{K+C}{4C}} + e^{-\frac{K+C}{6}} + o(1)$.*

*Proof.* Fix one round. By definition, peer $P_i$ is idle if (a) no connection request is sent towards $P_i$; (b) the connection request sent by $P_i$ is rejected. Let $P_i(\text{idle})$ be the prob. that peer $P_i$ is idle. Note that each peer $P_i$ will choose $C$ peers as neighbors uniformly at random from $\{P_1, P_2, \ldots, P_{i-1}\}$ and then one of them will be chosen uniformly at random for communication. Hence the connection request from $P_i$ will be sent to one of $\{P_1, P_2, \ldots, P_{i-1}\}$ that is chosen uniformly at random. By condition (a), for each $i \geq C + 1$, we have $P_i(\text{idle}) \leq \prod_{j=i}^{N-1} \left(1 - \frac{1}{j}\right) \approx \prod_{j=i}^{N-1} e^{-\frac{1}{j}} \approx \exp\left(-\ln \frac{N-1}{i-1}\right) = \frac{i-1}{N-1} \leq \frac{i}{N}$. Note that if $\deg(i) \leq K + C$, then any connection requests sent to $P_i$ will be accepted (with prob. 1). By Lemma 2, we have $E[\deg(i)] \leq \frac{K+C}{2}$ for all $i \geq e^{-\frac{K+C}{2C}} \cdot N$. By Chernoff Bound, we have $\Pr[\deg(i) > K + C] \leq e^{-\frac{K+C}{6}}$ for all $i \geq e^{-\frac{K+C}{2C}} N$. Hence condition (b) implies that for all $i \geq e^{-\frac{K+C}{4C}} N$, $P_i(\text{idle}) \leq 1 - (1 - e^{-\frac{K+C}{2C} + \frac{K+C}{4C}})(1 - e^{-\frac{K+C}{6}}) \leq e^{-\frac{K+C}{4C}} + e^{-\frac{K+C}{6}}$. Recall that for all $i \leq e^{-\frac{K+C}{4C}} N$, we have $P_i(\text{idle}) \leq e^{-\frac{K+C}{4C}}$, which finishes the proof. □

Note that Lemma 6 only gives a very loose upper bound on the fraction of idle peers and it is not hard to see from the proof that the exact fraction of idle peers should be much smaller than the upper bound proved above. Next we show how TTL helps us to avoid flooding of requests in our protocol.

**Lemma 7.** *Under our protocol, the number of requests at each peer is at most $h \cdot K \cdot R$.*

*Proof.* We call a request *disappears* if it is deleted since its TTL$= 0$, or it is merged with another request with a larger TTL value. Since TTL $= h$ for each newly generated request, a request must disappear $h$ rounds after it is generated.

Consider the set of request at a fixed peer $P_i$ at the end of some round $t$. By the above argument, we know that each request must be generated and transmitted to $P_i$ at some round after $t - h$. Since at most $K$ connections are build on $P_i$ and at most $R$ requests are sent to $P_i$, we know that the number of requests at $P_i$ is at most $h \cdot K \cdot R$. □

## IV. Performance Evaluation

We provide an accurate simulation and comparison in this section. We compare the performance with the lower bound proved in Theorem 1, and show how our protocol works by partially eliminating some important modules of the original model. Since we assume peers join the network incrementally at the beginning of the process, our experiment performs better than the lower bound in Theorem 1, which is based on the assumption that all $N$ peers join the network at the same time.

### A. Simulation Setting

We set $N = 2^{15} = 32768$ in all our experiments. A larger $N$ means that a larger population of peers can be served at the same time by our protocol. We set the number of chunks $M = 1000$. At the beginning only the server is in the network. We incrementally build the network by adding 10 new peers each round to the network until it reaches the maximum capacity $N$. We simulate all our experiments by running our protocol for 30000 rounds, which is sufficiently large to make sure that the (dynamic) network is stabled. We collect the fluency and latency of each peer when it leaves the network. More than 400000 data points are collected during our simulation. We also record information related to the network, including the number of requests, number of chunks transmitted, and so on.

As defined before, we set the fraction of random seekers to $\epsilon = 0.5$ and prob. of leaving during downloading to $\Pr(\text{exit}) = 1/4000$, which ensures that the fraction of peers that finish the watching is at least $1/e$. Each peer is assigned $C = 4$ different neighbors randomly when it joins the network. Each peer only initiates one communication and responds to a maximum number of $K = 4$ connection requests every round. We assume that it takes 4 rounds to consume (watch) one chunk. To reduce the network traffic, we set $R = 20$ to be the maximum number of requests each peer can send in each communication; $h = 10$ to be the initial TTL for each request, which is roughly the diameter of the network, by Lemma 1.

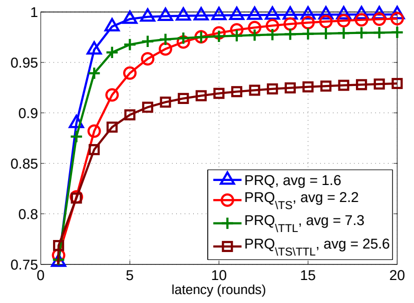
### B. PRQ Performance: Latency and Fluency

The objective of our protocol is to reduce the latency (Fig 1a) and ensure the fluency (Fig 1b) while only make use of constant number of connections and communications. To illustrate the importance of the fields TS and TTL in a request, we examine the performance of PRQ by partially disable some fields of a request. More details can be found in the figure 1c. We denote by $fr(*)$ the fraction of peers.

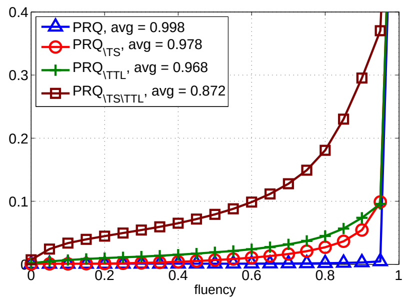
*1) PRQ$_{\backslash TS \backslash TTL}$:* A request in this case has no TS or TTL, and is only attached with the chunk ID it is requesting. As discussed in Section III, some issues will occur in this case: (1) flooding nature of request dissemination: a request at peer $P_i$ will only be deleted when $P_i$ receives the requested chunk. (2) race condition: the request generated earliest (which normally should be considered as "urgent" request) is not processed with a higher priority. As show in Fig 1c, it is unacceptable that more than 7% of the peers suffer from large latency ($\geq 20$).

*2) PRQ$_{\backslash TTL}$:* With the help of TS, it is easier to identify the requests generated earlier. However, without TTL, the number of requests at each peer may grow as large as $\Theta(M)$, which means that if a peer is under bad network condition (surrounded by peers with $\Theta(M)$ requests), then its latency can be really large. However, it can be seen that the latency and fluency are dramatically improved, compared to PRQ$_{\backslash \text{TS} \backslash \text{TTL}}$.

*3) PRQ$_{\backslash TS}$:* Compared to PRQ$_{\backslash \text{TTL}}$ the average latency is decreased from 7.3 to 2.2, while the average fluency increases from 0.968 to 0.978. However, the race condition hinders the performance of the whole network. It is interesting that PRQ$_{\backslash \text{TS}}$ outperforms PRQ$_{\backslash \text{TTL}}$ in improving the latency of peers under bad condition while PRQ$_{\backslash \text{TTL}}$ outperforms PRQ$_{\backslash \text{TS}}$ in improving the latency of peers under good condition. This is indeed natural: TS helps peers under good condition by identifying their requests as "urgent" while TTL prevents the requests of peers under bad condition from being submerged by enormous number of requests.



(a) CDF of latency

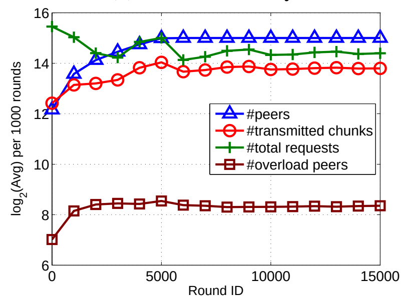


(b) CDF of fluency

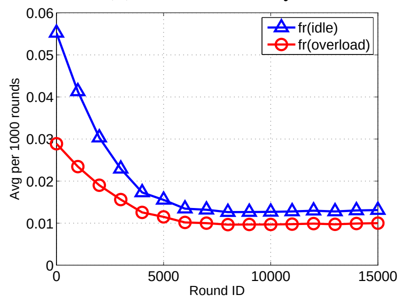
| Latency | =0 | ≤1 | ≤2 | ≤3 | ≤4 | ≤5 |
|---|---|---|---|---|---|---|
| PRQ | 2.5% | 75.2% | 89.0% | 96.3% | 98.5% | 99.3% |
| \TS | 2.5% | 75.9% | 81.7% | 88.2% | 91.8% | 93.9% |
| \TTL | 2.4% | 75.5% | 87.7% | 93.9% | 96.0% | 96.8% |
| \TS\TTL | 2.5% | 76.9% | 81.6% | 86.4% | 88.6% | 89.8% |

| Fluency | =1 | ≥0.99 | ≥0.98 | ≥0.97 | ≥0.96 | ≥0.95 |
|---|---|---|---|---|---|---|
| PRQ | 97.8% | 99.2% | 99.3% | 99.4% | 99.5% | 99.5% |
| \TS | 62.2% | 77.5% | 82.6% | 85.9% | 88.3% | 90.1% |
| \TTL | 85.3% | 87.7% | 88.6% | 89.3% | 89.9% | 90.4% |
| \TS\TTL | 45.2% | 53.3% | 56.6% | 59.0% | 61.1% | 63.0% |

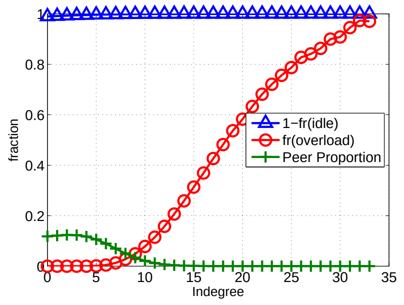
(c) Details on latency and fluency



(d) Network monitor



(e) w.r.t round ID



(f) w.r.t in-degree of peers

Fig. 1: Performance of PRQ, Number of Idle Peers and Overload Peers

*4) PRQ:* The average latency is 1.6 and average fluency is 0.998. According to figure 1c, $fr(l_u \leq 5) \approx 0.993$ and $fr(f_u \geq 0.95) \approx 0.995$, which means that only $0.7\% + 0.5\% = 1.2\%$ of the peers are less-satisfied. Since no protocol can achieve 0 average latency together with 1 fluency, at least $75.2\%$ of peers under our protocol achieve optimal latency ($\leq 1$) and at least $99.2\%$ of peers achieve optimal fluency ($\geq 0.99$). The above result suggests that our protocol achieves a great succeed in distributing requests evenly in the network.

## C. Network Communication Details

Each point in Fig. 1d is the average value of the corresponding variable over 1000 rounds and rescaled by $log_2(*)$.

*1) Number of requests:* It can be observed from Fig. 1d that after the first 5000 rounds, the average number of requests each peer maintains has dropped from 11 to less than one, which means that when the network is stabled, more requests are satisfied than generated. The observation implies that the total number of requests in the network is well controlled ($O(N)$), as theoretically analyzed in Lemma 7.

*2) Number of chunks:* The number of chunks transmitted every round is always proportional to the number of peers, as shown in Fig. 1d. We can see that roughly $\frac{N}{2}$ chunks are transmitted when there are $N$ peers in the network.

*3) Idle and overload peers:* As upper bounded in Lemma 6, the fraction of idle peers $fr(\text{idle})$, is extremely small in our protocol. Further more, as the number of peers in the network increases to $N$, $fr(\text{idle})$ decreases from 0.055 to 0.014, by Fig. 1e. According to Fig. 1f, $fr(\text{idle})$ remains almost the same for peers of different in-degree. Note that a peer is idle only if its connection request is sent to an overload peer (and hence rejected). Hence the fraction of overload peers $fr(\text{overload}) \leq fr(\text{idle})$, which is shown by Fig. 1e.

## References

[1] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *SIGCOMM*, 2008, pp. 375–388.

[2] K. Wang and C. Lin, "Insight into the p2p-vod system: Performance modeling and analysis." in *ICCCN*, 2009, pp. 1–6.

[3] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," in *INFOCOM*, 2011, pp. 694–702.

[4] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "Statistical modeling and analysis of p2p replication to support vod service," in *INFOCOM*, 2011, pp. 945–953.

[5] W. Wu and J. C. S. Lui, "Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation," in *INFOCOM*, 2011, pp. 1206–1214.

[6] D. Ciullo, V. Martina, M. Garetto, E. Leonardi, and G. L. Torrisi, "Stochastic analysis of self-sustainability in peer-assisted vod systems," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1539–1547.

[7] C. Zhao, X. Lin, and C. Wu, "The streaming capacity of sparsely-connected p2p systems with distributed control," in *INFOCOM*, 2011, pp. 1449–1457.

[8] C. Zhao, J. Zhao, X. Lin, and C. Wu, "Capacity of p2p on-demand streaming with simple, robust and decentralized control," in *INFOCOM*, 2013, pp. 2697–2705.

[9] B. Cohen, "Incentives build robustness in bittorrent," 2003.

[10] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of bittorrent-like protocols for on-demand stored media streaming," *ACM SIGMETRICS*, vol. 36, no. 1, pp. 301–312, 2008.

[11] W. Tang, X. Wu, and T. Hubert, "An incentive protocol for distributed dynamic p2p video-on-demand streaming," in *ICCCN*, 2014, pp. 1–6.

[12] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "A unifying model and analysis of p2p vod replication and scheduling," in *INFOCOM*, 2012, pp. 1530–1538.

[13] 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.

[14] X. Yang and G. De Veciana, "Service capacity of peer to peer networks," in *INFOCOM 2004*, vol. 4. IEEE, 2004, pp. 2242–2252.

[15] F. Mattern, "Virtual time and global states of distributed systems," *Parallel and Distributed Algorithms*, vol. 1, no. 23, pp. 215–226, 1989.

[16] F. Chen and X. Wu, "Perfect pipelining for streaming large file in peer-to-peer networks," in *IFIP TCS 2014*, 2014, pp. 27–38.