# Are a Few Neighboring Peers Good Enough?

Lili Zhong†, Jie Dai†, Bo Li†, Baochun Li‡, Hai Jin*
†*Hong Kong University of Science & Technology*, ‡*University of Toronto*
**Huazhong University of Science and Technology*

***Abstract*—Most peer-assisted media streaming systems have applied a design philosophy that uses a "mesh" topology of peers: each peer connects to a small number of neighboring peers, with which it exchanges state information periodically. Requests for media segments are made to these neighboring peers on-demand. Apparently, with such a "gossiping" design principle, peers do not make decisions based on global knowledge of the entire system, sacrificing system-wide efficiency. If a peer connects to more neighbors, the gap between local and global knowledge is mitigated; however, the *overhead* of communicating with all neighbors periodically is proportionally higher. In this paper, using theoretical analysis based on a system of difference equations, we show the surprising result that, once the number of neighbors exceeds a very small threshold (typically 5), the peer upload capacity is fully utilized. We demonstrate that such a threshold is not affected by the scale of system, and setting the number of neighbors beyond the threshold does not help alleviate challenges caused by peer churn. Our results imply that the communication overhead of typical "gossiping" protocols can very well be subdued and contained, by using a very small number of neighbors.**

## I. Introduction

The most competitive advantage of using peer assistance in large-scale media streaming systems is the scalability and robustness brought by engaging million of users as bandwidth contributors. With each peer fully utilizing its upload capacity to assist neighboring peers in a "mesh" topology, bandwidth load on dedicated streaming servers is effectively alleviated, and the entire system enjoys better resilience to peer arrivals and departures, commonly referred to as *peer churn*.

Most real-world peer-assisted streaming systems are designed with the "gossiping" design principle. Following this design principle, peers are organized in a mesh topology, with each peer connecting to a small number of "neighbors." Each peer exchanges state information (such as buffer availability maps) with its neighbors periodically, and sends requests for media segments to its neighbors on-demand. Due to the limited number of neighbors each peer has, decisions are made with local — rather than global — knowledge, and as such sacrificing system-wide efficiency. Though a peer can be connected with a larger number of neighbors, additional neighbors do bring additional costs in terms of *communication overhead:* state information will have to be exchanged with more neighbors, and the bandwidth required for such a periodic exchange increases proportionally as the number of neighbors increases.

One would naturally wonder how many neighbors are sufficient to maintain most of the advantages of peer assistance, including a consistent level of quality, the ability to scale to millions of peers, and robustness against peer churn. Surprisingly, this problem is not yet well understood and addressed. Boyd *et al.* [1] and Massoulie *et al.* [2] have modeled gossip protocols as a decentralized broadcasting problem. While studying the process of segment dissemination to participating peers, the maximum sustainable rate and minimum buffering delay of a system without peer churn and with *global knowledge* have been analyzed in [3] and [4], respectively. Feng *et al.* [5] have investigated the adverse effects without global knowledge. It has been shown that a performance gap exists between decentralized and centralized scheduling.

In contrast to previous work, in this paper, we focus on the number of neighbors required for "gossiping" protocols to work effectively in a mesh peer topology. With numerical results based on an analytical model consisting of a system of difference equations, we explore the pattern of distributing media segments in a mesh topology, without global knowledge and centralized scheduling. Our microscopic view on the number of neighbors leads to a number of surprising results. We show that a very small number of neighbors — for example, five — is sufficient to maintain a satisfactory level of streaming quality and full utilization of peer upload capacities. In addition, such a low threshold value is not affected by the scale of the system, and using more neighbors beyond the threshold does not help mitigate the negative effect caused by peer churn. For example, a peer with 5 neighbors experiences a similar degradation of streaming quality during peer churn as that with 20 neighbors. Our results imply that the communication overhead of typical "gossiping" protocols can be substantially contained and mitigated, by using a very small number of neighbors, with which state information is periodically exchanged.

## II. The Effect of Partial Knowledge in Systems Without Peer Churn

### A. System Model

Consider a peer-assisted streaming system that consists of one or multiple servers and $N$ peers. We define the *relative server upload capacity* $u_s$ as the ratio of the aggregate server upload capacity $U_s$ with respect to the video streaming rate $R$, *i.e.*, $u_s = U_s/R$. We assume that the bottleneck in peer-assisted streaming systems is the peer upload capacity [6]. Similarly, we can define the *relative peer upload capacity* $u_i$ of peer $i$ as the ratio of the peer upload capacity $U_i$ to $R$ ($i \in \{1, 2, 3, \ldots, N\}$), and the *average relative peer upload capacity* is denoted by $u_p$.

We consider a slotted system with the length equal to one segment playback time. It is assumed that, in each time slot, peer $i$ can send $\lfloor u_i \rfloor$ segments, plus one additional segment

TABLE I
KEY PARAMETERS IN THE SYSTEM MODEL.

| | |
|---|---|
| $N$ | Number of peers in the system. |
| $R$ | Streaming rate. |
| $U_i$ | Upload capacity of peer $i$. |
| $u_i$ | Relative upload capacity of peer $i$ ($= U_i/R$). |
| $u_p$ | Relative average peer upload capacity |
| $U_s$ | Server upload capacity. |
| $u_s$ | Relative server upload capacity ($= U_s/R$). |
| $n$ | Number of neighbors of a peer. |

with probability $u_i - \lfloor u_i \rfloor$. Similarly, the server can send $u_s$ segments per time slot. In each time slot, a peer randomly selects $n$ peers ($n$ is the *neighbor size*), and exchanges state information, such as buffer availability maps, with all its neighbors.

The notations introduced are summarized in Table I.

### B. Fundamentals of Partial Knowledge

We now introduce an analytical model to examine the influence of the neighbor size in a peer-assisted streaming system. The video segments, each with a unique sequence number, are distributed among peers in the order of playback. We assume that the *newest first* scheduling strategy is used in our basic model, as it was demonstrated to have the minimum performance gap with the optimal centralized scheme [5]. To keep things simple, suppose that the neighbor size $n$ is no less than the relative peer upload capacity $u_p$. Taking the fraction of peers that hold a certain segment as the main metric, we can formulate our problem in the proposition below.

Let $q_i(t)$ denote the number of peers that have segment $i$ at the beginning of time slot $t$. Denote by $m_i(t)$ the number of peers that would like to upload segment $i$ to its neighbors. Let $s_i(t)$ denote the number of copies a peer is able to upload.

*Proposition 1:* If $u_p$ is an integer, the evolution patterns for $\{m_i(t)\}$, $\{s_i(t)\}$ and $\{q_i(t)\}$ under the newest first scheduling strategy in a peer-assisted streaming system without peer churn can be approximated by the following difference equations:

$$q_{t+1}(t+1) = u_s \tag{1}$$

$$q_{t-i}(t+1) = q_{t-i}(t) + m_{t-i}(t)s_{t-i}(t) \tag{2}$$

$$m_{t-i}(t) = q_{t-i}(t)\prod_{j=0}^{i-1}\big(1-p_{t-j}(t)\big) \tag{3}$$

$$\text{where} \quad p_{t-i}(t) = q_{t-i}(t)/N$$

$$s_{t-i}(t) = u_p - \sum_{j=0}^{u_p-1}(u_p-j)\binom{n}{j}\big(1-p_{t-i}(t)\big)^j p_{t-i}(t)^{n-j} \tag{4}$$

$$i = 0, 1, 2, \dots t$$

*Proof:* In time slot $t$, the server distributes the newest segment $t+1$ to peers according to its upload capacity. Thus, $q_{t+1}(t+1) = u_s$. Now, turn to segment $t-i$. By definitions of $m_{t-i}(t)$ and $s_{t-i}(t)$, we know the number of peers that have segment $t-i$ at the beginning of the time slot $t+1$ would be increased by $m_{t-i}(t)s_{t-i}(t)$, which leads to (2).

We now turn to $m_{t-i}(t)$. Note that a peer will upload segment $t-i$ if and only if it is the newest segment in the buffer. Assume that for any peer, the event that it has segment $t-i$ is independent of the event that it has any other segment. The probability that segment $t-i$ is the newest for a peer can be approximated by $\prod_{j=0}^{i-1}\left(1-\frac{q_{t-j}(t)}{N}\right)$ under the assumption that segments are homogeneously distributed among peers, which results in (3).

For each of the $m_{t-i}(t)$ peers, let $X$ denote the number of its neighbors without segment $t-i$. Then each peer is able to upload to $\min(X, u_p)$ neighbors. Note that a neighbor has segment $t-i$ with probability $p_{t-i}(t) = q_{t-i}(t)/N$. Assume that this event is independent of the event that another neighbor does, then $X$ follows a binomial distribution with parameters $n$ and $p_{t-i}(t)$, i.e. $X \sim \mathrm{B}(n, p_{t-i}(t))$. Thus,

$$\begin{aligned}
s_{t-i}(t) &= E(\min(X, u_p)) \\
&= \sum_{j=0}^{n}\min(j, u_p)\binom{n}{j}\big(1-p_{t-i}(t)\big)^j p_{t-i}(t)^{n-j} \\
&= \sum_{j=0}^{u_p-1} j\binom{n}{j}\big(1-p_{t-i}(t)\big)^j p_{t-i}(t)^{n-j} \\
&\quad + \sum_{j=u_p}^{n} u_p\binom{n}{j}\big(1-p_{t-i}(t)\big)^j p_{t-i}(t)^{n-j} \\
&= u_p - \sum_{j=0}^{u_p-1}(u_p-j)\binom{n}{j}\big(1-p_{t-i}(t)\big)^j p_{t-i}(t)^{n-j}
\end{aligned}$$

■

Proposition 1 with Eq. (12) indicates that the neighbor size $n$ plays an important role in the distribution pattern of segments. It will be thoroughly demonstrated in Sec. II-D.

Now we prove a basic principle in Proposition 1 that every segment follows the same evolution of $q_i(t)$ over time.

*Theorem 1:* The evolution pattern for $\{q_i(t)\}$ in Proposition 1 satisfies that

$$q_{t-i}(t) = q_{t-i+1}(t+1) \quad (i = 0, 1, 2, \dots, t) \tag{5}$$

*Proof:* This can be proved by induction.

(1) If $i = 0$, according to (1), we have
$\forall t,\ q_t(t) = q_{t+1}(t+1) = u_s$.

(2) Assume that for each $j, s$ $\big(0 \le j \le i-1, 0 \le s \le t-1\big)$, it is satisfied that $q_{s-j}(s) = q_{s-j+1}(s+1)$. We would like to prove $q_{t-i}(t) = q_{t-i+1}(t+1)$. From (2), we know

$$q_{t-i+1}(t+1) = q_{t-i+1}(t) + m_{t-i+1}(t)s_{t-i+1}(t)$$

We analyze $q_{t-i+1}(t)$, $m_{t-i+1}(t)$, $s_{t-i+1}(t)$ respectively.

$$
\begin{aligned}
q_{t-i+1}(t) &= q_{(t-1)-(i-1)+1}\big((t-1)+1\big) \\
&= q_{(t-1)-(i-1)}(t-1) = q_{t-i}(t-1) \\
m_{t-i+1}(t) &= m_{t-(i-1)}(t) = q_{t-i+1}(t)\prod_{j=0}^{i-2}\big(1-p_{t-j}(t)\big) \\
&= q_{t-i}(t-1)\prod_{j=0}^{i-2}\big(1-p_{(t-1)-j}(t-1)\big) \\
&= m_{t-i}(t-1)
\end{aligned}
$$

$$
\begin{aligned}
& s_{t-i+1}(t) \\
&= s_{t-(i-1)}(t) \\
&= u_p - \sum_{j=0}^{u_p-1}(u_p-j)\binom{n}{j}\big(1-p_{t-i+1}(t)\big)^j p_{t-i+1}(t)^{n-j} \\
&= u_p - \sum_{j=0}^{u_p-1}(u_p-j)\binom{n}{j}\big(1-p_{t-i}(t-1)\big)^j p_{t-i}(t-1)^{n-j} \\
&= s_{t-i}(t-1)
\end{aligned}
$$

Therefore, we have

$$q_{t-i+1}(t+1) = q_{t-i} + (t-1)m_{t-i+1}(t)s_{t-i+1}(t) = q_{t-i}(t)$$

Combining (1) and (2), we obtain $q_{t-i}(t) = q_{t-i+1}(t+1)$. ■

Since for all $0 \le i \le t$, $q_{t-i}(t) = q_{t-i-1}(t-1) = \cdots = q_0(i)$, the evolution of $q_0(t)$ over time $\{q_0(0), q_0(1), \ldots q_0(k)\}$ equals to the sequence $\{q_k(k), q_{k-1}(k), \ldots, q_0(k)\}$. Because $q_i(k)/N$ is the probability that a peer have segment $i$ in time slot $k$, we can observe the expectation of a peer's buffer in a particular time slot, from the evolution of $q_l(t)/N$ (for all $l$).

### C. Non-Integral Relative Peer Upload Capacity

In this subsection, we examine the case when $u_p$ is not integral.

*Proposition 2:* If $u_p$ is not an integer, the evolution patterns for $\{s_i(t)\}$ under the newest first strategy in a P2P streaming system without peer churn can be approximated by the following difference equations:

$$
\begin{aligned}
s_{t-i}(t) = {} & \overline{s}_{t-i}\big(t, \lfloor u_p \rfloor\big)\cdot\big(\lfloor u_p+1\rfloor - u_p\big) \\
& + \overline{s}_{t-i}\big(t, \lfloor u_p \rfloor + 1\big)\cdot\big(u_p - \lfloor u_p\rfloor\big)
\end{aligned} \tag{6}
$$

where

$$\overline{s}_{t-i}(t,x) = x - \sum_{j=0}^{x-1}(x-j)\binom{n}{j}\big(1-p_{t-i}(t)\big)^j p_{t-i}(t)^{n-j} \tag{7}$$

*Proof:* When $u_p$ is an integer, $s_{t-i}(t) = \overline{s}_{t-i}(t, u_p)$ in (7) is equivalent to that in (4). So we only prove (6).

When $u_p$ is not integral, a peer can be considered to have an upload capacity of $\lfloor u_p \rfloor$ with probability of $\lfloor u_p+1\rfloor - u_p$, and $\lfloor u_p \rfloor + 1$ segments with probability of $u_p - \lfloor u_p \rfloor$. Summing them up, we obtain (6). ■

### D. Numerical Analysis of Systems Without Churn

We now discuss the effect of the neighbor size by examining the distribution of video segments based on Proposition 1. Since each segment evolves along the same pattern according to Theorem 1, we keep track of the fraction of peers that hold segment 100 in Fig. 1 (on page 5) as an illustration. Fig. 1 reveals that a very small neighbor size slows down the propagation of an individual segment due to a limited scope of content availability information obtained from neighbors. This phenomenon may be referred to as *the effect of partial knowledge* in "gossiping" protocols. More specifically, only approximately $68\%$ of peers obtain the given segment with a neighbor size of 1, in sharp contrast to the low skip rate with a neighbor size of more than 5. This result has shown that the effect of partial knowledge — caused by a very small number of neighbors (less than 5) — can prevent the system from taking full advantage of peer uploading capacities. However, this effect is significantly subdued with a slight increase in the number of neighbors, say, to more than 5.

We next consider how the scale of the system affects the number of neighbors required. As shown in Fig. 1, each segment eventually reach a stable portion of peers, defined as the *successful fetching ratio* of continuous video segments. Fig. 2 illustrates the variance of peer successful fetching ratios with different system scales and neighbor sizes. We observe that the successful fetching ratio is insensitive to the scale of the system, when the neighbor size increases to larger values (more than 5). However, the variance of successful fetching ratios under different system scales becomes significant once we use neighbor sizes lower than a threshold value, say 5. This reveals that using a very small number of neighbors will prevent the system from scaling up. These results prove that *the effect of partial knowledge is not relevant to the scale of the system*, if the number of neighbors a peer has is beyond a small threshold, say, 5.

We further investigate segment distribution under different peer upload capacities. As shown in Fig. 3, the successful fetching ratio achieves the optimum value if we increase $u_p$ from 1 to 1.2, when the neighbor size is $> 5$. This result suggests that *higher peer upload capacity helps to maintain the sustainable streaming performance, by alleviating the impact from partial knowledge*. This observation is consistent with the measurement results from real world system, Coolstreaming+ [7], in which while larger number of partners increases the probability in finding capable parents, such improvement is marginal when the neighbor size reaches a critical value. This, however, is perhaps more relevant in peer-assisted on-demand streaming systems, in which the neighboring candidates are more restricted.

### E. Random Scheduling Strategy

We now consider a different scheduling algorithm, the *random* scheduling strategy, and see if a similar observation can be obtained. With random scheduling, a downstream peer randomly selects a missing segment in its buffer, without any

preference, when requesting from a neighbor. Let $B$ denote the size of a peer's buffer.

*Proposition 3:* If the *random* scheduling strategy is adopted, the evolution pattern for $\{m_i(t)\}$ in a peer-assisted streaming system without peer churn can be approximated by the following difference equation:

$$m_{t-i}(t) = \frac{q_{t-i}(t)}{r(t)}, \text{ where } r(t) = \frac{1}{N}\sum_{i=0}^{B-1} q_{t-i}(t) \quad (8)$$

*Proof:* Assume a homogenous distribution of segments among peers, then $r(t)$ is the expectation of the number of segments owned by a peer. In this case, segment $t-i$ has a probability of $1/r(t)$ to be uploaded. Overall, the number of peers that would like to upload the segment is approximately $q_{t-i}(t)/r(t)$. ∎

Observed from Fig. 4, the fractions of peers holding a given segment under both scheduling strategies essentially converge to the same stable values, approximately $68\%$ and $96\%$ under neighbor sizes of 1 and 10, respectively. This demonstrates that our observations made so far — that a small number of neighbors is good enough to alleviate the effect of partial knowledge — are not limited to a specific scheduling strategy.

## III. Peer Churn

We next consider the scenario with peer churn. Intuitively, a larger neighbor size can alleviate the fluctuation caused by peer churn, by maintaining the network topology [8] and helping the video segment distribution. However, contrary to the intuition, we will demonstrate that an increase of the neighbor size does *not* help the propagation of segments in a system suffering from peer churn. Without loss of generality, we analyze a system with one round of peer joining and leaving. We define the *churn rate* as the fraction of newly joined peers (*new peers*) among all the peers in the system, which consist of new peers and existing peers remaining in the system (*old peers*) after peer departure. Since segments are expected to follow different distribution patterns for new and old peers, we analyze them separately.

*Lemma 1:* Let $U^0$, $U^1$ denote the number of old peers and new peers that a peer connects to, respectively, and $X^0(t)$, $X^1(t)$ denote the number of requests for segment $t-i$ that a peer receives from its old and new neighbors in time slot $t$, respectively. Then

$$\Pr\left(X^\kappa(t) = j | U^\kappa = u\right) = \binom{u}{j}\left(1-p^\kappa_{t-i}(t)\right)^j p^\kappa_{t-i}(t)^{u-j} \quad (9)$$

$$\kappa \in \{0,1\}$$

The proof of Lemma 1 is omitted since it is similar to the proof of (4).

Let $q^0_{t-i}(t)$ and $q^1_{t-i}(t)$ denote the number of old and new peers that have segment $t-i$, respectively. Define $m_{t-i}(t)$ as the number of peers that plan to upload segment $t-i$. Let $s^0_{t-i}(t)$ and $s^1_{t-i}(t)$ denote the number of copies that a peer is able to upload to old and new peers, respectively.

*Proposition 4:* If $u_p$ is an integer, the evolution patterns for $\{m_i(t)\}$, $\{s^\kappa_i(t)\}$ and $\{q^\kappa_i(t)\}$ $(\kappa \in \{0,1\}, i = 0,1,\ldots,t_0)$ under *the newest first strategy* in a peer-assisted streaming system with a peer churn rate of $b$ can be approximated by the following.

For $t \le t_0$, $\{q^0_i(t)\}$ follows Proposition 1.

For $t > t_0$, we have the following difference equations:

$$q^\kappa_{t-i}(t+1) = q^\kappa_{t-i}(t) + m_{t-i}(t)s^\kappa_{t-i}(t) \quad (10)$$

$$m_{t-i}(t) = \sum_{\kappa\in\{0,1\}} q^\kappa_{t-i}(t)\cdot\prod_{j=0}^{i-1}\left(1-p^\kappa_{t-j}(t)\right) \quad (11)$$

$$\begin{aligned} s^\kappa_{t-i}(t) = {} & \bar{s}^\kappa_{t-i}\left(t, n-\lfloor bn\rfloor, \lfloor bn\rfloor\right)\cdot\left(\lfloor bn\rfloor + 1 - bn\right) \\ & + \bar{s}^\kappa_{t-i}\left(t, n-\lfloor bn\rfloor - 1, \lfloor bn\rfloor + 1\right)\cdot\left(bn - \lfloor bn\rfloor\right) \end{aligned} \quad (12)$$

where $p^\kappa_{t-i}(t) = q^\kappa_{t-i}(t)/N$

$$\begin{aligned} & \bar{s}^\kappa_{t-i}(t, u^0, u^1) \\ = {} & \sum_{k=0}^{u^\kappa}\binom{u^\kappa}{j}\left(1-p^\kappa_{t-i}(t)\right)^j p^\kappa_{t-i}(t)^{u^\kappa - j} \\ & \left(k\sum_{j=0}^{u_p-k}\Pr\left(X^{1-\kappa}(t)=j|U^{1-\kappa}=u^{1-\kappa}\right)\right. \\ & \left. + \sum_{j=u_p-k+1}^{u^{1-\kappa}}\left(\Pr\left(X^{1-\kappa}(t)=j|U^{1-\kappa}=u^{1-\kappa}\right)\frac{ku_p}{k+j}\right)\right) \end{aligned}$$

$\kappa \in \{0,1\},\ i = 0,1,2,\ldots t.$

*Proof:* The proofs of (10) and (11) are similar to those of (2) and (3).

A peer will connect to $bn$ new peers in expectation under the assumption that each peer connects to the same number of old peers and new peers, respectively. Therefore, we prove (12) in case of $\kappa = 0$ in the following two cases.

First, suppose $bn$ is an integer. If $X^0(t) + X^1(t) \le u_p$, all the requests can be satisfied. Otherwise, the uploader selects $u_p$ neighbors at random, thus the expected number of selected old ones is $X^0(t)u_p/\left(X^0(t)+X^1(t)\right)$. Therefore given $X^0(t) = k$, the number of old neighbors that successfully download the segment is $k\cdot\sum_{j=0}^{u_p-k}\Pr\left(X^1(t)=j|U^1=bn\right)$ $+\sum_{j=u_p-k+1}^{bn}\left(\Pr\left(X^1(t)=j|U^1=bn\right)\cdot\frac{ku_p}{k+j}\right)$.

Taking the distribution of $X^0(t)$ in Lemma 1 into account, we have $s^0_{t-i}(t) = \bar{s}^0_{t-i}(t, n-bn, bn)$, which is equivalent to (12) for an integral $bn$.

Second, when $bn$ is not an integer, we suppose that a peer will connect to at least $\lfloor bn\rfloor$ new peers, and one more with probability $bn - \lfloor bn\rfloor$. Combining cases of $bn$ and $bn+1$ new neighbors, we have Eq. (12).

To summarize, (12) holds for any value of $bn$. ∎

Fig. 5 compares the evolution patterns of a particular segment (14) in a system without churn ($q_i(t)$) and with peer churn occurring in time slot 20. We consider the evolution patterns for both old peers ($q_{old}(t)$) and newly joined peers

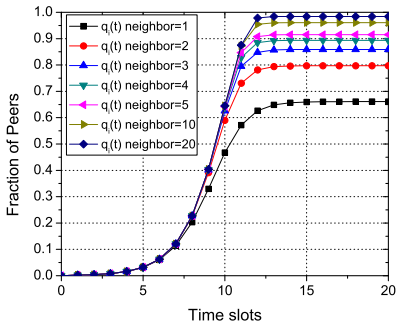


Fig. 1. The fraction of peers holding segment 100 versus time slots, under different neighbor sizes. The system size $N$ is set to 10000. Others are set as $u_s = 20$, $u_p = 1$.

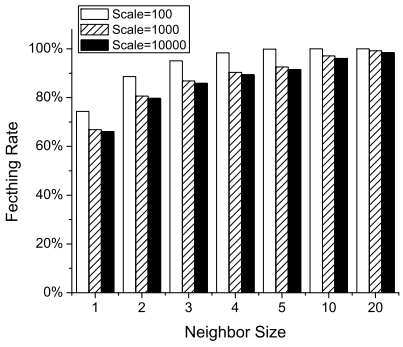


Fig. 2. A comparison of successful fetching ratios under different scales and neighbor sizes. The neighbor size varies from 1 to 20. The system scale is set to 100, 1000 and 10000. Others are set as $u_s = 20$, $u_p = 1$.

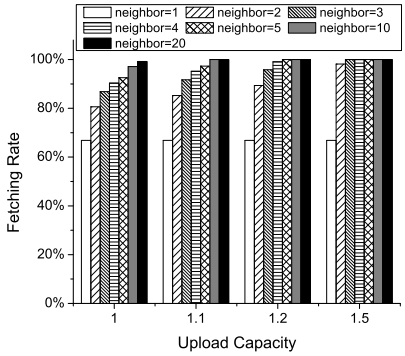


Fig. 3. Successful fetching ratios versus various peer upload capacities. The value of $u_p$ varies under different settings. The neighbor size varies from 1 to 20. Others are set as $u_s = 20$, $N = 1000$.

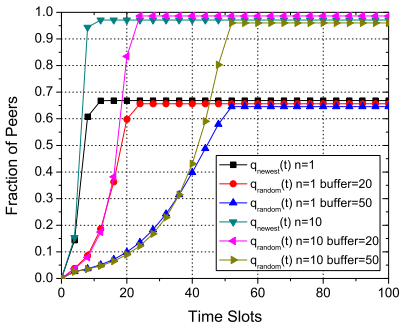


Fig. 4. A comparison of the evolution pattern of segments under newest first strategy and the random forwarding. The buffer size under the random strategy varies from 30 to 50. Others are set as $u_s = 20$, $u_p = 1$, $N = 1000$, $n = 10$ and 1

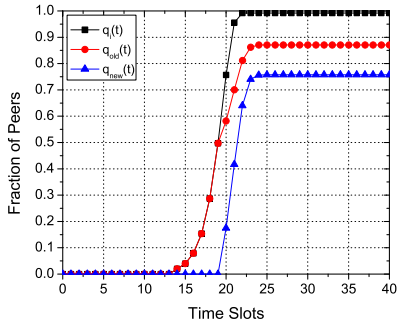


Fig. 5. Systems without churn versus systems suffering from peer churn in terms of the segment evolution. The churn happens in time slot 20. The observed segment is delivered by the server in time slot 14. We set the churn rate to 0.5. Others are set as $u_s = 20$, $u_p = 1$, $N = 1000$, $n = 10$.

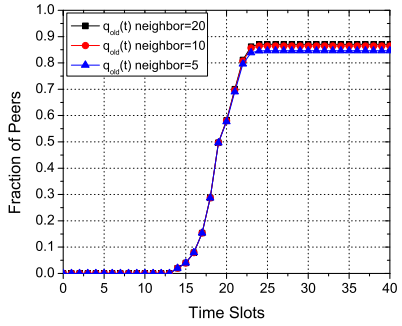


Fig. 6. A validation of segment evolution of old peers in case of peer churn. The peer churn happens in time slot 20. The observed segment is delivered by the server in time slot 14. The neighbor size varies from 5 to 20. We set the churn rate to 0.5. Others are set as $u_s = 20$, $u_p = 1$, $N = 1000$.

($q_{new}(t)$) under peer churn. The churn rate is set to a relatively large value of 0.5. As shown in Fig. 5, $q_i(t)$ and $q_{old}(t)$ share an identical increasing trend before time slot 20. As the peer churn occurs, the marginal rate of increase with respect to the fraction of old peers holding the segment decreases sharply, in contrast to the behavior in systems without churn. This is caused by the fact that newly joined peers would compete with old peers for the segment by issuing download requests to their connected neighbors. Furthermore, the newly joined peers are not yet capable of providing the otherwise needed uploading capacity.

However, contrary to our intuition, Fig. 6 reveals that an increase in the neighbor size can not alleviate the temporarily degraded performance caused by peer churn. In fact, peers experience a similar degree of degradation with respect to the playback continuity under neighbor sizes of 5, 10 and 20.

## IV. Conclusion and Future Work

In this paper, we have presented an analytical model to gain important insights into the number of neighbors required for peer-assisted streaming systems that employ "gossiping" protocols. Surprisingly, We have observed a very small threshold for the required number of neighbors that leads to the full utilization of peer upload capacities, as well as the scale-free property of the system. When the system is experiencing peer churn, performance degradation can hardly be mitigated by merely adjusting the neighbor size. These results serve as important guidelines when the number of neighbors is determined in real-world peer-assisted streaming systems.

## References


[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip Algorithms: Design, Analysis and Applications," in *Proc. of IEEE INFOCOM*, 2005.

[2] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized Decentralized Broadcasting Algorithms," in *Proc. of IEEE INFOCOM*, 2007.

[3] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, 2007.

[4] Y. Liu, "On the Minimum Delay Peer-to-Peer Video Streaming: How Realtime Can It Be?" in *Proc. of ACM Multimedia*, Sep. 2007.

[5] C. Feng, B. Li, and B. Li, "Understanding the Performance Gap between Pull-based Mesh Streaming Protocols and Fundamental Limits," in *Proc. of IEEE INFOCOM*, 2009.

[6] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems," in *Proc. of ACM Internet Measurement Conference (IMC 2006)*, Oct. 2005.

[7] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications," in *Proc. of IEEE INFOCOM*, 2008.

[8] A. Allavena, A. Demers, and J. E. Hopcroft, "Correctness of a Gossip Based Membership Protocol," in *Proc. of ACM PODC*, Jul. 2005.