ViVUD: Virtual Server Cluster based View-Upload Decoupling for Multi-Channel P2P Video Streaming Systems

Chao Liang and Yong Liu
Department of Electrical and Computer Engineering
Polytechnic Institute of New York University
Brooklyn, New York, 11201, USA
Email: {cliang@photon.poly.edu, yongliu@poly.edu}

***Abstract*—Despite the success to deliver increasingly large number of channels to millions of users, the current multi-channel P2P video streaming systems still suffer several fundamental performance problems, such as large start-up delays and poor performance for unpopular channels. To alleviate the impact of channel churn and resource imbalance, the View-Upload Decoupling (VUD) [1] P2P streaming design decouples peer downloading and uploading, and enables cross-channel resource sharing. However, VUD incurs upload bandwidth overhead and distribution swarm management cost. It is also challenging to adapt VUD distribution swarms in extreme peer churn scenarios, such as flash-crowd. In this paper, we propose ViVUD, a Virtual Server Cluster based VUD design. In ViVUD, a virtual server cluster consisting of bandwidth-rich peers is provisioned to improve the streaming quality of each channel. A virtual server cluster provides stable video feeds to boost peers newly joining a channel to reduce their start-up delays. To enable cross-channel bandwidth sharing, following the VUD design, virtual server clusters for unpopular channels are formed by bandwidth-rich peers watching popular channels. Through analysis and simulations, we show that, compared with the original VUD design, ViVUD incurs less upload bandwidth overhead, has lighter management requirement, achieves lower channel start-up delays, and adapts faster to flash crowds.**

I. Introduction

In these years, several large-scale commercial P2P video streaming systems have been deployed, including Coolstreaming [2], PPLive [3]. Recent measurement studies have verified that hundreds of thousands of users can be simultaneously participating in these systems [3]. Nearly all P2P video systems offer multiple channels. And future user-generate-video streaming systems will likely have thousands if not millions of live channels.

A common practice in P2P video streaming today is to organize peers viewing the same channel into a swarm, with peers in the same swarm redistributing video chunks exclusively to each other. We refer to such a design as *isolated-channel* or more simply as *ISO* P2P streaming systems. However, recent measurement studies [3] have identified several fundamental performance problems for isolated-channel systems: *large start-up delay* and *poor small-channel performance*. Current start-up delays are typically on the order of 10-60 seconds, which is clearly undesirable as users are accustomed to delays under 3 seconds in current cable and satellite television systems. Furthermore, current P2P streaming systems generally provide inconsistent and poor performance to channels with a small number of peers.

Recently, a new approach to multi-channel P2P streaming is proposed, referred as *View-Upload Decoupling (VUD)* [1]. VUD strictly decouples what a peer uploads from what it views, bringing stability to multi-channel systems and enabling cross-channel resource sharing. In VUD, each peer is assigned to one or more channels, with the assignments made *independently of what the peer is viewing*. For each assigned channel, the peer helps distribute the channel. This has the effect of creating a semi-permanent *distribution swarm* for each channel, which is formed by peers responsible for uploading that channel. Although VUD can effectively address the performance issues due to channel churn and resource imbalance in ISO systems, it still suffers from the following issues:

- *Upload bandwidth overhead*. VUD introduces upload bandwidth overhead since each peer now needs to upload to its assigned channel distribution swarm as well as to peers outside the distribution swarm but are the viewers of the channel.
- *Distribution swarm management cost*. All viewers of a channel download the video from the channel's VUD distribution swarm. A VUD swarm needs to be constantly monitored and maintained to serve the time-varying viewing demand.
- *Sluggish adaption to flash crowds*. VUD distribution swarms are built to be semi-permanent. It is challenging to adapt VUD swarms when there is a sudden increase in viewing demand for a channel, such as flash crowds.

In this paper, we propose a new cross-channel streaming design, ViVUD, which improves VUD by the introduction of virtual server clusters. In ViVUD, instead of complete

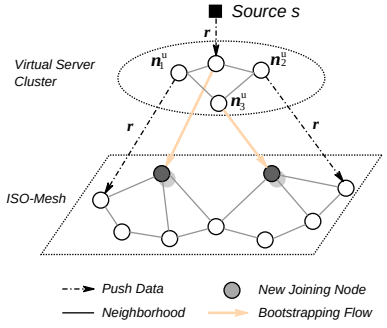


Fig. 1. An illustration of the ViVUD streaming framework.

view-upload decoupling on all peers, most peers upload and download the video they are viewing in the ISO fashion. The VUD design is only applied to a small fraction of bandwidth-rich peers, which are assigned to form semi-permanent virtual server clusters, one for each channel. Instead of streaming video to all viewers of a channel, the virtual server cluster is provisioned to improve the quality of the ISO streaming between the viewers. Specifically, a virtual server cluster provides stable video feeds to peers newly joining a channel to reduce their start-up delays. To improve streaming quality on existing peers, a virtual server cluster acts as a video proxy and "amplifies" the bandwidth of the video source server by actively pushing packets into the ISO mesh. To address the upload deficiency in unpopular channels, virtual server clusters for unpopular channels are formed by bandwidth-rich peers viewing popular channels. Figure 1 illustrates the ViVUD streaming framework for a single channel.

We propose a set of algorithms to implement ViVUD streaming framework in Section III. The dimensioning of virtual server clusters are studied in Section IV. With queuing network model, we compare the universal streaming probability of ViVUD with that of VUD and ISO design in Section V. Finally, through simulations in Section VI, we demonstrate that, compared with the original VUD design, ViVUD incurs less upload bandwidth overhead, has lighter management requirement, achieves lower channel start-up delays, and adapts faster to flash crowds.

II. Related Work

Most of the previous research on P2P system design focuses on single isolated channel, without much consideration for multi-channel optimization. There are only few related work in the multi-channel setting. Server bandwidth provisioning algorithms are proposed to adjust the supply of server bandwidth among channels in streaming [4] and file sharing applications [5]. Inter-overlay cooperation mechanisms [6] are proposed to share resources among channels. Decentralized strategies are investigated to resolve the conflicts among coexisting streaming overlays [7]. The above works mostly consider the balance of bandwidth resources. They do not directly address the channel churn and channel resource imbalance problems in multi-channel streaming systems. Recently, the view-upload decoupling streaming design [1][8] addresses those problems by strictly decoupling peer viewing and uploading. Our work differs in that, we can also resolve the performance problems as VUD does, but the enhanced approach can be readily implemented with far less overhead, and can adapt faster to flash crowd.

III. System Design

A. ViVUD Architecture Overview

As illustrated in Figure 1, in ViVUD, there are three major streaming components for a channel: the channel source server, VUD Virtual Server Cluster (VSC) formed by bandwidth-rich peers, and ISO streaming mesh formed by viewers of the channel. Peers in VSC form a mesh and collaboratively download video from the channel source server. Due to the rich bandwidth availability on VSC peers, VSC becomes a video proxy that "amplifies" the bandwidth of the original source server. VSC improves the quality of video streaming to channel viewers in two ways. To reduce the start-up delay, when a peer joins a channel, it directly downloads the first x seconds of video in the download window quickly from the channel's VSC. This can greatly reduces the start-up delay for peers newly joining the system, or switching from other channels. To improve the streaming quality of existing peers, VSC pushes copies of content to the main ISO mesh of viewers to accelerate the video distribution.

B. VSC Formation and Management

In the original VUD design, the channel's distribution swarm streams the video to all the viewers. While in ViVUD, a VSC is only responsible for bootstrapping new peers and assisting streaming in ISO mesh. Therefore, VSCs have lighter streaming workload than VUD swarms. The number of peers needed in a VSC is smaller than that in a VUD swarm. Consequently, the upload bandwidth overhead and management cost of ViVUD can be made much lower than VUD.

In practice, popular channels with a large number of viewers have stable and high upload bandwidth availability, and are more resilient to bandwidth fluctuations caused by peer churn [3][9]. Peers in popular channels rarely suffer from bandwidth deficiency. On the other hand, bandwidth availability in unpopular small channels are very dynamic. As a result, the streaming quality in unpopular channels is not consistent [3][10]. To address the resource imbalance problem between popular and unpopular channels, VSCs only employ bandwidth-rich peers watching popular channels. Cross-channel resource sharing is enabled by assigning peers from popular channels to VSCs of unpopular channels. Peers watching unpopular channels are helped by stable data feeds from VSCs, and achieve resilience against bandwidth deficiency. To further reduce upload bandwidth overhead, the VSC of a popular channel can be formed by stable bandwidth-rich peers watching the same channel. It is also possible to adopt the substream-based swarm design [1] for VSC.

We resort to a tracker-based solution to manage VSCs. The tracker maintains a pool of bandwidth-rich peers from popular channels. VSC of each channel has a targeted size (We will discuss the VSC dimensioning issue in the next section). For the initial construction, the tracker assigns peers from the pool to each VSC to reach its target size. The tracker constantly monitors the operations of VSCs. Whenever a VSC experiences bandwidth deficiency, either due to some peers leave the VSC, or there is a sudden increase in the number of viewers for the channel, the tracker augments the VSC by allocating more peers to it.

C. Data Sharing Strategies

VSC plays two roles in the video streaming of a channel. First, it serves as the video proxy and source amplifier for the ISO mesh. (If allowed by its upload capacity constraint, the

source server s can directly stream additional data to the ISO mesh.) A small number of VSC peers are reserved to push stable video feeds to the ISO mesh. Second, VSC bootstraps the newly joining peers to enhance their start-up performance. When a peer joins a channel, it contacts the tracker for a list of VSC peers for the channel. The new peer identifies a VSC peer as its helper and establishes a connection with it. The helper bootstraps the new peer by pushing the first x seconds of the video content in its download window. If the helper can upload to the new peer at a rate higher than the streaming rate, the bootstrapping video can be uploaded in $y < x$ seconds. The connection with the helper is terminated after the helper finishes the bootstrapping task. While bootstrapped, the new peer joins the channel's ISO mesh, and downloads the video after the first x seconds from its neighbors there.

VSC adaptively adjusts the bandwidth allocation between video pushing to ISO mesh and bootstrapping new peers. If the actual new peer arrival rate is lower than the expected rate, idle peers without bootstrapping tasks can also push video content to the ISO mesh to accelerate the data distribution. On the other hand, when the new peer arrival rate is higher than the expected rate, VSC becomes under-provisioned for new peer bootstrapping. There are two ways to respond. The VSC can lower the bootstrapping workload for each new peer by reducing x, the length of bootstrapping video, so that each new peer will spend less time in the bootstrapping stage. It opens up some space for VSC to bootstrap more new peers. Or when newly joining peers cannot be accommodated by the VSC, they would be directly assigned to the ISO mesh without going through the bootstrapping stage. In that way, they could still achieve the start-up delay performance of the ISO design.

IV. Dimensioning the Virtual Server Cluster

In this section we investigate how to dimension VSCs for efficient cross-channel sharing. Suppose in a multi-channel P2P video system, there are J channels. For channel j, the source server upload bandwidth is v_j and the streaming rate is r_j. The distribution of peer viewing time is arbitrary. Let u_i be the upload rate of peer i. The bandwidth on a VSC peer is consumed in three different tasks: (I) within VSC, peers collaboratively retrieve a copy of the video from the source through P2P upload-download; (II) a VSC peer uploads video to a new peer to bootstrap its start-up performance; (III) a VSC peer pushes video to the ISO mesh to accelerate the content distribution. The per-peer bandwidth consumption for the first task is roughly equal to the video streaming rate (or the substream rate for substream-based VSC). Only a few VSC peers are sufficient for the third task. So, we would mainly focus on dimensioning VSC for the bootstrapping task.

1) VSC Bootstrapping Capacity. Suppose for channel j, the VSC uploads to newly joining peers the first x seconds of content at rate βr_j in y seconds, i.e., $x = y\beta$. Let X_j denote the random variable of the number of new peers joining the channel within y seconds. We define the capacity δ of a VSC as the number of new peers that the VSC can simultaneously bootstrap. To achieve a bootstrapping probability $\mathcal{C}$, the capacity of VSC for channel j should satisfy

$$P(X_j < \delta_j) \geq \mathcal{C}. \tag{1}$$

The aggregate bootstrapping upload bandwidth required is $\delta_j \beta r_j$.

2) VSC Provisioning for Cross-channel Sharing. Unpopular channels borrow bandwidth-rich peers from popular channels to form their VSCs. To reduce view-upload decoupling bandwidth overhead, we can adopt substream-based design for VSCs for unpopular channels, i.e., a VSC peer only downloads and uploads one substream. Furthermore, the VSC of a popular channel would be formed by bandwidth-rich peers viewing the same channel. Next, we discuss how to conduct cross-channel sharing between popular channels and unpopular channels.

Let r_j^s be the rate of the sth substream in channel j and $\mathcal{N}_j^s$ be the set of VSC peers handling substream s for channel j. For an unpopular channel $j \in J^U$, to satisfy the bootstrapping requirement in Equation (1), we obtain

$$\sum_{i \in \mathcal{N}_j^s} (u_i - r_j^s) \geq \delta_j \beta r_j^s. \tag{2}$$

In addition to the bootstrapping bandwidth, VSC of an unpopular channel also addresses the bandwidth deficiency of ISO mesh. For this purpose, an additional amount P_j of bandwidth is needed for pushing video to secure the streaming performance in ISO mesh. If channel j is divided into S^j substreams with homogeneous rate, and the VSC only accepts one type of peer with upload capacity u_p, the bandwidth overhead ratio is thus $1/S^j$ and the number of VSC peers needed by channel j is $\frac{\delta_j \beta r_j + P_j}{u_p - r_j/S^j}$.

Definition 1: The *net cross-channel resource flow* σ_j of channel j is defined as the net amount of outgoing/incoming bandwidth to/from other channels without overhead.

Suppose the relative popularity of channel j is ρ_j. The cross-channel resource flow vector σ is determined as follows. For popular channels, we have

$$\sigma_j = -\frac{\rho_j}{\sum_{i \in J \setminus J^U} \rho_i} \sum_{i \in J^U} \frac{\delta_i \beta r_i + P_i}{u_p - r_i/S^i} u_p, \forall j \in J \setminus J^U. \tag{3}$$

The allocation heuristic behind the rule is that *the channel with more popularity should share more resources to help the unpopular ones*. For unpopular channels, the net cross-channel resource flow is $\sigma_j = \delta_j \beta r_j + P_j$.

V. Modeling and Analysis

To compare the performance of ViVUD with VUD and ISO design, we apply the closed queuing network models and formulate the multi-channel P2P streaming system in the same way as [8]. Under the same closed queuing network model, we focus on the case of a fixed number of peers, modeling the systems with always-on devices or low peer churn rate.

1) *Queueing network model.* Let $\lambda = (\lambda_1, \cdots, \lambda_J)$ be the unique probability distribution of arrival rate that satisfies $\lambda = \lambda \mathbf{P}$, where $\mathbf{P}$ is the channel switching matrix. Let $1/\mu_j$ denote the expected amount of time a peer continuously views channel j and define $\rho_j = \lambda_j/\mu_j$. Note that λ_j is the relative arrival

rate into channel j, and ρ_j is the relative channel popularity. Small channels have relatively low values of ρ_j. We normalize the ρ_j's so that $\rho_1+\cdots+\rho_J = 1$. Let M_j be a random variable denoting the number of peers viewing channel j. Each of n peers is viewing some channel. Because the total number of peers viewing channels is fixed at n, we have $M_1+\cdots+M_J = n$. Thus, the random variables $M_1, \ldots, M_J$ are dependent. By viewing the system as an infinite-server Jackson queuing network [11], with each channel being a node in the network, and each peer as a customer which sojourns at node j for a random amount of time with mean $1/\mu_j$, we immediately arrive at the following result:

Lemma 1: For any $m_1, \ldots, m_J$ with $m_1 + \cdots + m_J = n$, we have

$$P(M_1 = m_1, \ldots, M_J = m_J) = n! \frac{\rho_1^{m_1}}{m_1!} \cdots \frac{\rho_J^{m_J}}{m_J!} \quad (4)$$

Note that $(M_1, \ldots, M_J)$ has a multinomial distribution. It follows directly from the lemma that $E[M_j] = n\rho_j$ and that M_j has the binomial distribution

$$P(M_j = m_j) = \frac{n!}{m_j!(n-m_j)!} \rho_j^{m_j} (1-\rho_j)^{(n-m_j)}. \quad (5)$$

With the distribution of the number of viewers in each channel, we now determine the distribution of the number X_j of newly joining peers to be bootstrapped. Let y_j be the bootstrapping period for channel j. Given y_j is short, we ignore the rare case that a newly joining peer leaves channel j before the bootstrapping is over, and assume all new peers are bootstrapped for y_j seconds. Each channel could be modeled by a tandem of two queues, as illustrated in Fig. 2. The first queue has the deterministic service time of y_j, while the average service time for the second queue is $1/\mu_j - y_j$. The number X_j of peers in bootstrapping stage corresponds to the number of viewers in the first queue. The relative popularity of the first queue for channel j can be calculated as $\rho_j^1 = y_j \mu_j \rho_j$. Hence, we can derive the distribution of X_j based on Lemma 1. And we can further determine the dimensioning of VSCs, and the net cross-channel resource flow σ.

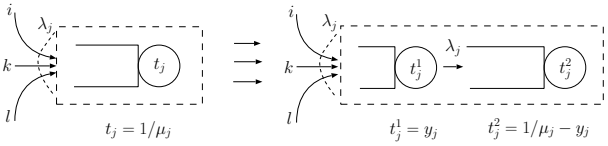


Fig. 2. Transformation of channel queue.

Let $\mathcal{M}_j$ be the set of peers viewing channel j. The probability of system-wide universal streaming of ViVUD is

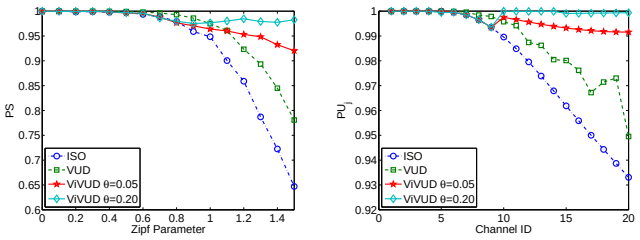
$$PS = P(v_j + \sum_{i \in \mathcal{M}_j} u_i + \sigma_j \geq M_j r_j,\ j = 1, \ldots, J)$$

where $\mathcal{M}_j$ is a random set and $M_j = |\mathcal{M}_j|$. We can calculate this probability using Monte Carlo methods and importance sampling [12]. We repeatedly generate samples $\{\mathcal{M}_1, \ldots, \mathcal{M}_J\}$, evaluate the event in the above probability (as 0 or 1) and take averages. Suppose there are two classes of peers: n^l peers with low upload rate u^l; and n^h peers with high upload rates u^h. For a given channel j in ViVUD, the probability of universal streaming is given by

$$\begin{aligned} PU_j &= P(v_j + \sum_{i \in \mathcal{M}_j} u_i + \sigma_j \geq M_j r_j) \\ &= P(v_j + u^h M_j^h + u^l M_j^l + \sigma_j \geq (M_j^h + M_j^l) r_j) \\ &= \sum_{M_j^h=0}^{n^h} \sum_{M_j^l=0}^{n^l} I(v_j + u^h m_j^h + u^l m_j^l + \sigma_j \geq \\ &\quad (m_j^h + m_j^l) r_j) P(M_j^h = m_j^h) P(M_j^l = m_j^l), \end{aligned}$$

where $P(M_j^h = m_j^h) = \binom{n^h}{m_j^h} \rho_j^{m_j^h} (1-\rho_j)^{n^h - m_j^h}$ and $P(M_j^l = m_j^l) = \binom{n^l}{m_j^l} \rho_j^{m_j^l} (1-\rho_j)^{n^l - m_j^l}$.

2) *Numerical results.* In the simulation setting, there are $1,800$ peers and 20 channels. Each channel has the same rate r and the channel popularity follows a Zipf distribution. There are two classes of peers with upload bandwidth $u^l = 0.2r$ and $u^h = 3r$. These two classes of peers have the equal distribution. For the dimensioning of ViVUD, we set $\mathcal{C} = 0.8$ and $\beta = 1$. VSC peers download the full video stream. In VUD, the streaming in each channel is divided into 10 substreams. A channel with the average number of viewers less than 30 is treated as an unpopular channel. We assume the bootstrapping time for an unpopular channel j is set to be $y_j = \theta/\mu_j$, $\theta < 1$. Fig. 3(a) shows ViVUD can achieve



(a) PS (b) PU_j

Fig. 3. The universal streaming probability of PS and PU_j.

higher probability of system-wide universal streaming under different channel popularity distributions. Fig. 3(b) plots the probability of universal streaming for each channel when the Zipf parameter is 1.5. We can observe that the unpopular channels in ViVUD are more resilient to bandwidth fluctuation than VUD and ISO, the cross-channel sharing does not have much negative impact on the popular channels, thanks to the relatively small resource demands from unpopular channels. Furthermore, ViVUD incurs much less overhead, which is less than 1% even when $\theta = 0.2$, while the overhead of VUD is 10% with 10 substreams in each channel.

VI. PERFORMANCE EVALUATION

In this section, we conduct extensive packet-level simulations to evaluate the performance of the proposed strategies.

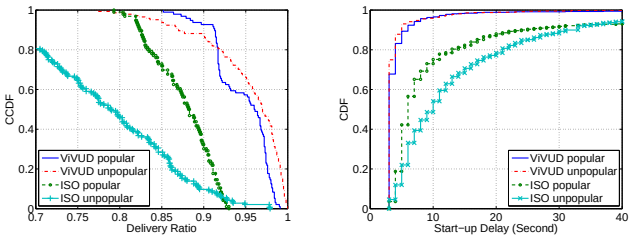
A. Simulation Setting

To compare the performance of different multi-channel streaming design, we implement an event-driven P2P streaming simulator which can simulate packet-level transmissions and end-to-end latency among peers. In our simulation, it is assumed that the bottleneck only happens at the edge of networks, and all the participating peers have enough download bandwidth. The video streaming rate is 400kbps. The source server upload bandwidth is set to be 1Mbps as default for each channel. Three types of peers are assigned with upload bandwidth 1.6Mbps, 384kbps and 128kbps. The corresponding fractions of them are $(0.15, 0.51, 0.34)$, and the resource index of the system is almost 1.2.

To complement the analysis based on closed-queuing network, we simulate an open multi-channel system. There are 15 channels and the popularity of the channels follows Zipf distribution. Peers arrive at the system according to a Poisson process. Peer viewing time in a channel follows a Weibull distribution. After a peer finishes viewing a channel, it leaves the system with certain probability, or switches to another channel with a probability proportional to the channel popularity. The average number of active peers in the system is $2,000$.

B. Simulation Results

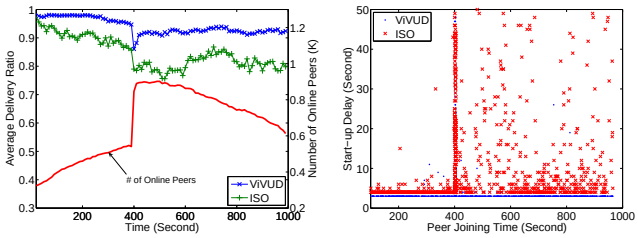
1) Multi-channel Performance: VSCs are formed by the class of peers with 1.6Mbps upload bandwidth. Each VSC is provisioned with twice of the required bootstrapping bandwidth. The bootstrapping time for new peers is 2 seconds. The start-up delay is counted as the time that a new peer needs to fill up at least 70% of the first 5 seconds of buffer since it joins the channel. Figure 4(a) plots the CCDF of the video delivery ratio performance. ViVUD greatly improves streaming quality for unpopular channels. Almost 80% unpopular channels have delivery ratio above 90%, while the delivery ratio is around 70% in ISO design. Furthermore, Figure 4(b) shows that ViVUD can achieve lower start-up delay with nearly 90% peers having start-up delay less than 5 seconds.



(a) Delivery Ratio (b) Start-up Delay

Fig. 4. Multi-channel performance.

2) Robustness against flash crowds: We compare the single channel performance of ViVUD and ISO under flash crowd scenario. In this simulation, the peer arrival follows the Poisson distribution with mean $\lambda = 1$. The peer lifetime is distributed with Weibull distribution parameters (600,2). In ViVUD, each VSC is provisioned with 6 peers for bootstrapping, and 5 peers reserved for pushing to ISO mesh. Between time 400 and 405, there is a batch arrival of peers at a rate of 60 peers per second. Figure 5(a) shows the evolution of number of online peers and delivery ratio of both schemes. We can observe that ViVUD can provide higher delivery ratio than ISO. Figure 5(b) presents the start-up delay of peers with different joining time. We can observe that peers cannot get satisfactory start-up delay performance during the short flash crowd period. Other than that period, ViVUD can always bootstrap new peers well and allow them to achieve lower start-up delay than ISO.



(a) Delivery Ratio (b) Start-up Delay

Fig. 5. Single channel performance under flash crowd.

VII. Conclusion

In this paper, we proposed a virtual server cluster based VUD design (ViVUD) to address the performance problems in the original VUD and ISO design. ViVUD employs virtual server clusters to bootstrap newly joining peers, and enables cross-channel sharing by assigning bandwidth-rich peers from popular channels to VSCs of unpopular channels. Through modeling, analysis and simulation, we demonstrated that, compared with the original VUD design, ViVUD has lighter management requirement, achieves lower channel start-up delays, and adapts faster to flash crowds.

References

[1] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-Upload Decoupling: A Redesign of Multi-Channel P2P Video Systems," in *Proceedings of IEEE INFOCOM*, 2009.

[2] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," in *Proceedings of IEEE INFOCOM*, 2008.

[3] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, 2007.

[4] C. Wu, B. Li, and S. Zhao, "Multi-channel live p2p streaming: Refocusing on servers," in *Proceedings of IEEE INFOCOM*, 2008.

[5] R. S. Peterson and E. G. Sirer, "Antfarm: Efficient Content Distribution with Managed Swarms," in *Symposium on Networked System Design and Implementation (NSDI)*, 2009.

[6] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," in *Proceedings of IEEE INFOCOM*, 2006.

[7] C. Wu, B. Li, and S. Zhao, "Strategies of conflict in coexisting streaming overlays," in *Proceedings of IEEE INFOCOM*, 2007.

[8] D. Wu, Y. Liu, and K. Ross, "Modeling and Analysis of Multi-Channel P2P Live Video Systems," in *Proceedings of IEEE INFOCOM*, 2009.

[9] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proceedings of IEEE INFOCOM*, 2007.

[10] Z. Liu, C. Wu, B. Li, and S. Zhao, "Why are peers less stable in unpopular p2p streaming channels?" *IFIP Networking*, 2009.

[11] F. P. Kelly, *Reversibility and Stochastic Networks*. Chichester: Wiley, 1979.

[12] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods (second edition)*. New York: Springer-Verlag, 2004.