International Journal of Web Services Research

October-December 2014, Vol. 11, No. 4

Table of Contents

Special Issue on New Techniques of Services Computing

Guest Editorial Preface

iv *Jia Zhang, Carnegie Mellon University, Pittsburgh, PA, USA*
Hanhua Chen, Huazhong University of Science and Technology, Wuhan, China

Research Articles

1 **Automatic Construction of Service Network based on OpenCyc**
Xiaocao Hu, School of Computer Science and Technology, Tianjin University, Tianjin, China
Zhiyong Feng, School of Computer Science and Technology, Tianjin University, Tianjin, China
Shizhan Chen, School of Computer Science and Technology, Tianjin University, Tianjin, China

24 **Regularity and Variability: Growth Patterns of Online Friendships**
Lun Zhang, Department of Journalism & Science Communication, University of Chinese Academy of Sciences, Beijing, China
Jonathan J. H. Zhu, Department of Media and Communication, City University of Hong Kong, Kowloon, Hong Kong

37 **Improving Recommendation Accuracy and Diversity via Multiple Social Factors and Social Circles**
Yong Feng, Ministry of Education, Chongqing University, Chongqing, China
Heng Li, Ministry of Education, Chongqing University, Chongqing, China
Zhuo Chen, Ministry of Education, Chongqing University, Chongqing, China

52 **An Integrated Framework for Semantic Service Composition using Answer Set Programming**
Yilong Yang, Department of Computer and Information Science, University of Macau, Macau, China
Jing Yang, College of Computer Science and Technology, Guizhou University, Guiyang, China
Xiaoshan Li, Department of Computer and Information Science, University of Macau, Macau, China
Weiru Wang, Department of Computer and Information Science, University of Macau, Macau, China

67 **Improve Distributed Client Lifecycle Control in ShadowStream**
Junhua Yan, Huazhong University of Science and Technology, Wuhan, China
Chen Tian, Huazhong University of Science and Technology, Wuhan, China
Jingdong Sun, Huazhong University of Science and Technology, Wuhan, China
Hanzi Mao, Huazhong University of Science and Technology, Wuhan, China

The *International Journal of Web Services Research* is indexed or listed in the following: ABI/Inform; ACM Digital Library; Bacon's Media Directory; Burrelle's Media Directory; Cabell's Directories; Compendex (Elsevier Engineering Index); CSA Illumina; Current Contents®/Engineering, Computing, & Technology; DBLP; DEST Register of Refereed Journals; Gale Directory of Publications & Broadcast Media; GetCited; Google Scholar; INSPEC; Journal Citation Reports/Science Edition; JournalTOCs; Library & Information Science Abstracts (LISA); MediaFinder; Norwegian Social Science Data Services (NSD); PubList.com; Science Citation Index Expanded (SciSearch®); SCOPUS; The Index of Information Systems Journals; The Standard Periodical Directory; Thomson Reuters; Ulrich's Periodicals Directory; Web of Science

Improve Distributed Client Lifecycle Control in ShadowStream

Junhua Yan, Huazhong University of Science and Technology, Wuhan, China

Chen Tian, Huazhong University of Science and Technology, Wuhan, China

Jingdong Sun, Huazhong University of Science and Technology, Wuhan, China

Hanzi Mao, Huazhong University of Science and Technology, Wuhan, China

ABSTRACT

ShadowStream is a novel Internet live streaming system that integrates performance evaluation as an intrinsic capability. An essential component in ShadowStream is distributed lifecycle control mechanism, which assigns each client a virtual arrival/lifetime to create a particular scenario to evaluate the performance of streaming system. The original design focuses on utilizing stable streaming viewers in physical world to guarantee the accuracy of ShadowStream, which, on the other hand, significantly limits the scale of the experiment. The authors' research develops a novel distributed client lifecycle control to get rid of restrictions caused by the limited number of stable viewers in live-testing streaming networks. The core idea of their research is to match the desired experimental scenario with real viewers' behavior in physical world. The result demonstrates that with the authors' methodology, the scale of experiments can be doubled.

Keywords: Lifetime, Live Streaming, PCE, ShadowStream, User Behavior

1. INTRODUCTION

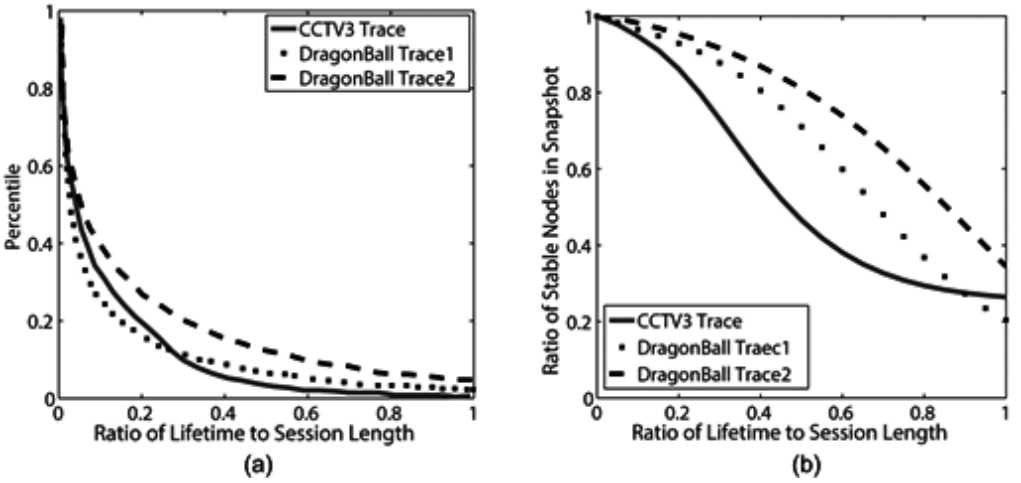
ShadowStream (Chen&Richard, 2012, pp. 347-358) is a novel Internet live streaming system that integrates performance evaluation as intrinsic capability. It introduces a novel *production-CDN-experiment (PCE)* streaming machine layout to protect real viewers' quality-of-experience (QoE) in experiment, at the same time gets accurate results.

An essential component in ShadowStream system is lifecycle control. In general, it lets a production viewer participate in the experiment by assigning arrival time and lifetime to emulate a virtual client's arrive and departure events. To distinguish it with clients' *real* behavior time in physical world, we call it *virtual* arrival/departure time.

To achieve this, we have introduced distributed mechanism in the process of control. When appointed a specific behavior scenario, orchestrator will send relevant parameters to testing clients to let them locally compute their arrival/life time for testing. In case of a

DOI: 10.4018/IJWSR.2014100105

Figure 1. The general situation about lifetime distribution and ratio of stable nodes per snapshot in physical world. (a) CCDF of life time distribution in different traces. (b) Ratio of stable nodes to All Nodes per snapshot.



client quits the experiment ahead of its virtual departure time unexpected, which we call *early-quitted* client, orchestrator will choose another viewer as a replacement and duplicate its status to the client.

However, *early-quitted* clients do have a negative influence on accuracy, since the process of replacement cannot be totally seamless. Thus we merely sort out stable viewers in physical world to perform distributed control for a specific behavior scenario to minimize the impact of replacement, which, on the other hand, has imposed restrictions on the scale of experiment in ShadowStream. Since in general cases stable clients are usually too small a group to be effectively utilized in production channel, which has been proved in Figure 1 (Wang&Liu, 2008). Figure 1(a) indicates that the majority of viewers just stay in channel for a quite short period in physical world, and if consider a client as stable when its lifetime exceed 40% of the observed session, it takes up only 5% to 18% of the whole viewers in different traces. Furthermore, Figure 1(b) explains the percentage of stable clients in a per-snapshot view in channel, and it is clear that there exist only 54% to 90% stable clients in a snapshot under the circumstances above.

In this paper, a novel distributed client lifecycle control is developed to get rid of restrictions caused by limited number of stable viewers in live-testing streaming networks. And dedicates to increasing real viewers' utilization level in physical world and decreasing replacement times in the process of experiment.

The major challenge in the course of experiment control in live testing platform is about the real viewers. As a live testing system, ShadowStream is designed to orchestrate desired experimental scenarios from production viewers, without disturbing their quality of experience. Other than clients in a traditional testing platform, production viewers in live testing cannot be controlled. Furthermore, we are not allowed to interfere in their own behaviors (*e.g.*, arrive, depart), or change their behaviors casually. That is to say, before sending a command to a client, guarantees should be made to ensure that the command which determines client's act in the experiment will not be in conflict with its real behavior at some point in the future.

The core idea is to match the desired experimental scenario with viewers' real behavior in physical world to scale up the experiment.

Moreover, we make a further survey about taking advantage of a certain group of stable viewers in physical world to achieve the desired scenario by letting them rejoin the testing under the specific situation.

The remainder of this paper is organized as follows: Section 2 introduces the background of our research about ShadowStream and experiment orchestration in detail. And Section 3 describes the basic idea about predicting each viewer's residual lifetime in physical world and matching it with the specific scenario in experiment. Then Section 4 explicitly explains the implementation of orchestration and relevant algorithms to control testing clients' behavior in experiment before we carefully evaluate its performance and testify the validity in Section 5.

1.1. Related Work

Live streaming (Chang&Wang, 2009; Li&Keung, 2008; Yin&Chiu, 2007) is a major Internet application in our daily life and online streaming has dominated the traffic on today's Internet (Cisco, 2012). And the key capability to guarantee that live streaming networks provide reliable performance is to subject them to large-scale, realistic performance evaluations, which is among the most desired and the most difficult to achieve. Thus many developers have turned their way to theoretical modeling (Bonald&Massoulie, 2008; Zhou&Chiu, 2007; Kumar&Liu, 2007), simulation (Magharei&Rejaie, 2007), or testbed/lab testing (Banerjee&Bhattacharjee, 2002; Castro&Druschel, 2003; Picconi&Massoulie, 2008). However, all of those existing methods fail to live up to the testing requirements considering their limitation either in scale or in realism.

Shadowstream attempts to gets rid of the limitation caused by tradition evaluation platform considering the fact that live streaming systems updated without going through realistic, large-scale evaluations may operate at sub-optimal states, and often do not achieve performance expectations formed at small-scale lab settings. And it integrates evaluation into production live streaming systems, which provides experiment scales that are not possible in any existing testing platforms such as VINI (Bavier&Feamster, 2006), PlaneLab (Chun&Culler, 2003), or Emulab (White&Lepreau, 2002).

2. SHADOWSTREAM BACKGROUND

2.1. The PCE Design

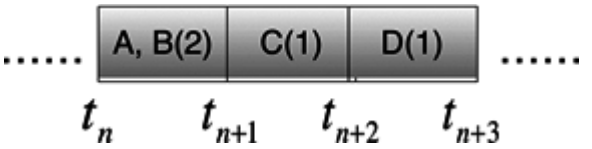
ShadowStream is a novel Internet live streaming system that integrates performance evaluation as an intrinsic capability. It is a hybrid P2P-CDN piece based streaming system, where a live streaming client downloads and uploads streaming data in units of pieces. In ShadowStream, a self-complete set of algorithms to download and upload pieces is called a streaming machine or a machine for short.

ShadowStream needs to address two major challenges. The first is protection: since real viewers are used, live testing needs to protect the real viewers' quality of experience from the performance failures of experimental systems. The second is orchestration: live testing needs to orchestrate desired experimental scenarios from production viewers, without disturbing their quality of experiences. ShadowStream applies a novel *Experiment* → *Validation* → *Repair* scheme to achieve both protection and transparent orchestration simultaneously.

The key idea is to handle each downloading task of a piece in a temporal sequential pattern. To reveal the true performance of *experiment*, ShadowStream assigns the task of downloading a piece first to *experiment* alone. If *experiment* cannot download it by its playpoint, *rCDN* takes over the responsibility and try to "repair" by download from CDN. If the repair of *rCDN* fails, *production* shall try to download it as a final protection. We evaluate the performance of the testing VoD streaming network by the piece missing ratio of *experiment*.

A streaming hypervisor is introduced to inform streaming machines of pieces which they should download respectively. Conceptually the total downloading range spanning from real playpoint to sourcepoint is divided into

Figure 2. Streaming machine sliding download window in ShadowStream: (a) at $t = 100$; (b) at $t = 101$



three parts by *production*, *rCDN* and *experiment* sequentially. Each part is referred as the task window of corresponding streaming machine. The playpoint and sourcepoint advance in time along with the task windows and the piece missed by the right task window becomes the downloading task of the left one. As shown in Figure 2, *rCDN* takes over the responsibility to download piece 90 if *experiment* fails to download it from time $t = 100$ to $t = 101$.

2.2. Test Orchestration

To deal with real viewers in production channel, it has introduced novel, local orchestration algorithms to orchestrate desired experimental scenarios in addition to the PCE streaming machine layout, and the major components of experiment orchestration in ShadowStream is shown in Figure 3.

Experiment orchestrator takes charge of notifying a large number of clients in real-time about their time to join and leave a testing scenario according to the defined arrival rate function $\lambda(t)$ and lifetime function L to generate the specified testing behavior scenario. And the orchestration time line is illustrated in Figure4, from which we can observe that other than the real behavior time in physical world, each testing client poses a *virtual* arrival/departure time in experiment.

A striking feature of experiment orchestration in ShadowStream is having proposed an effective, distributed algorithm where each client can *locally* decide and control its arrival/departure time. The theoretical basis to support this algorithm is the theorem from Cox and Lewis (Cox&Lewis, 1966), which proves that we can generate event times by drawing random numbers independently according to the same distribution function without executing global computation.

Although ShadowStream has overcome the limitations in scale and in realism to some extent, when compared with traditional evaluation methods, there still exists room for more improvements. Since our previous work about experiment orchestrator in ShadowStream just randomly notifies each client's behavior time by a distributed algorithm, without considering its *real* behavior in physical world, which may lead to a situation in Figure 5 that a client with a relatively shorter lifetime in physical world is expected to stay much longer in experiment, while the other has *virtually* departed the testing way ahead of its *real* departure time. And we define those clients in the latter situation as *early-departed* clients in experiment. Besides, we used to focus on testing client's departure time in experiment, which later has been proved to be unintuitive when taking viewers' real behavior into consideration since it is based

Figure 3. Components of experiment orchestration in ShadowStream system

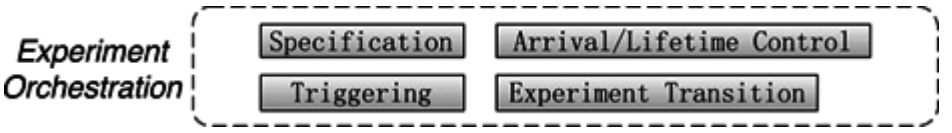
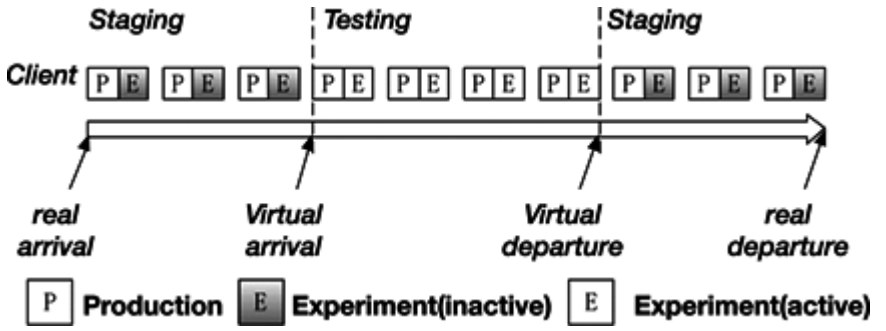


Figure 4. ShadowStream orchestration time line: real arrival, virtual arrival, virtual departure, real departure



on their residual lifetime in physical world that we make a prediction about the real behavior scenario.

Thus we decide to draw our attentions to lifetime distribution in the process of orchestration, and try to match each client's *virtual* lifetime in testing with its *real* residual lifetime in physical world. In general, clients with a relatively longer lifetime in physical world should also have a longer virtual lifetime in experiment.

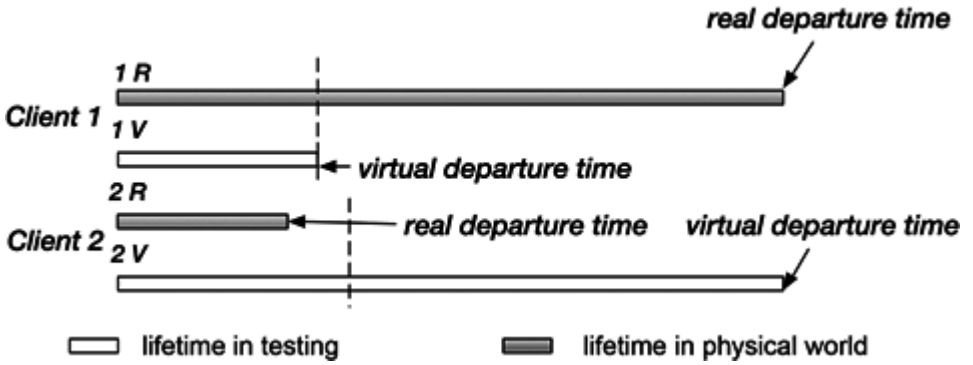
3. BASIC IDEA

Considering that real viewers' behavior in physical world mainly impacts the actual distribution of lifetime in experiment, the key to achieve the specific experimental scenario is matching it with real viewers' residual lifetime in testing. Thus it requires us to be aware of each testing viewer's *real* residual lifetime and match it with the desired distribution.

3.1. Expected Residual Lifetime

It is known that, in a peer-to-peer system, the node lifetime can be approximated by a Pareto distribution. To simplify the presentation, we assume that a client's lifetime is merely dependent on its arrival time in physical world, ignoring the influence of video quality for now. That is to say, the longer a client has stayed in channel, the longer it would stay in the future (Bishop&Rao, 2006, pp. 1-13).

Figure 5. An awkward situation may appear in experiment control: the mismatch between viewers' real departure time and virtual departure time



At current time t, the Cumulative Distribution Function $F_i(t)$ of client i's lifetime can be described by $Pareto(I)(\sigma,\alpha)$ with positive parameter σ, α, and the distribution function is:

$$F_{T_i}(t)=\begin{cases}1-\left(\frac{t}{\sigma}\right)^{-\alpha}, t\geq\sigma\\ 0, t<\sigma\end{cases} \quad (1)$$

where $\sigma = t - t_i'$, t_i' stands for the time when client i arrives. Thus σ means the total time it has spent in the channel by current time t. And the residual lifetime probability function P of client i can be described as follows:

$$P(T_i > t)=\left(\frac{t}{\sigma}\right)^{-\alpha}, t\geq\sigma \quad (2)$$

where T_i is client i's lifetime.

According to Swartz (1973), the mean residual lifetime function of a client that has survived to at least time t is $V(t)\equiv E\{T-t \mid T\geq t\}$. To be more specifically:

$$R(t)=1-F_T(t)=P(T>t) \quad (3)$$

$$V(t)=\frac{1}{R(t)}\int_t^{\infty}R(x)dx \quad (4)$$

Based on (1) (3) (4), the average remaining lifetime is:

$$V(t)=\frac{t}{\alpha-1}, \alpha>1 \quad (5)$$

Thus for client i who has survived to at least time t in the channel, its probable departure time $D_i(t)$ can be presented as follows.

$$D_i(t)=t_i'+t+V_i(t) \quad (6)$$

3.2. Process of Matching

Now we give an example to illustrate the process of sorting and matching in experiment orchestration. In the example, we present the desired lifetime distribution in a certain period in Figure 6(a) by defining the expected number of clients in each timescale, while in physical world there exists four real viewers (A,B,C,D) whose average residual lifetime is distributed in the specific testing duration as described in Figure 6(b). Comparing the desired and the real lifetime distribution, we select three testing clients in physical world and present their residual lifetime and the actual distribution after sorting in experiment in Figure 6(c), which indicates there still exists gaps with the desired scenario. Thus it requires lifetime control to make client D depart the testing ahead of its real departure time to achieve the desired lifetime distribution as illustrated in Figure 6(d).

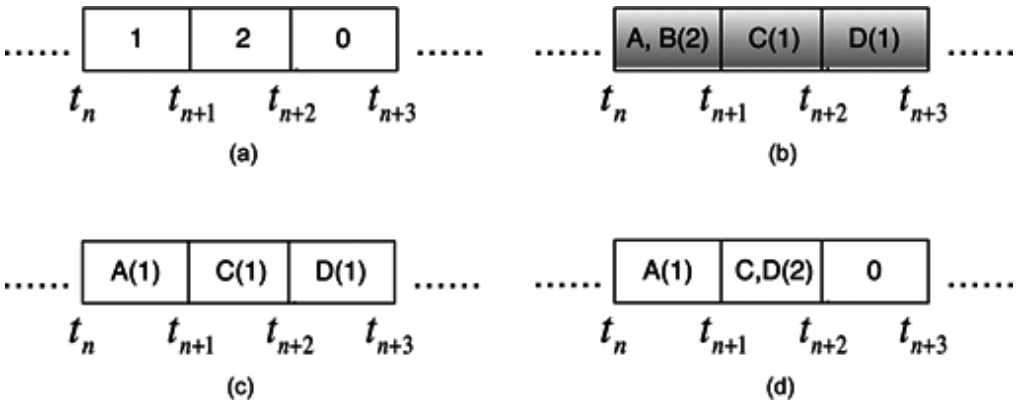
4. ALGORITHM DESIGN

In this section, we explicitly introduce the implementation of experiment orchestration and describe relevant algorithms when taking real viewers' residual lifetime into account. To further improve the utilization level, we make those *early-departed* viewers can rejoin to the experiment.

4.1. Triggering Condition

Our previous work merely focuses on whether the number of stable clients in physical world at any time t satisfies the desired behavior scenario when it comes to the triggering condition. And it has greatly limited the scale of testing as well as influenced the actual behav-

Figure 6. Process of sorting and matching in experiment orchestration. (a) Desired lifetime distribution in testing. (b)Real lifetime distribution in testing after sorting. (c) Actual lifetime distribution in testing after sorting. (d) Actual lifetime distribution in testing after lifetime control.



ior scenario, since knowing that a client is stable alone is not enough to predict its behavior in near future. Thus, we decide to take each client's lifetime function L_x and the global expected lifetime distribution $\phi(t)$ into account to redefine the triggering condition. The triggered test starting time is referred to as t_{start}, and the test runs from t_{start} to $t_{start} + t_{\exp}$.

To estimate the lifetime distribution in a certain period, we divide testing duration $[0, t_{\exp}]$ into $\frac{t_{\exp}}{\Delta t}$ timescales, and denoted as $t_0, t_1, ..., t_i, t_{i+1}, ..., \frac{t_{\exp}}{\Delta t}$, where Δt is the minimum processing time. At any time t, orchestrator predicts the number of active clients whose residual lifetime V in physical world satisfies the condition $\{V \mid t_n < V \le t_{n+1}\}, n = 0,1,..., \frac{t_{\exp}}{\Delta t}$, by a simple extension of the autoregressive integrated moving average (ARIMA) method (Wu&Li, 2008) that uses both recent testing channel states and the past history of it, and let $Predict(t + t_n)$ present the predicted value. To obtain current testing channel states, the orchestrator gathers channel state (arrivals and departures) from clients' low cost UDP reports. The expected number of clients $Exp(t_n)$ whose lifetime ranges from t_n to t_{n+1} in experiment can be computed as follows:

$$Exp(t_n) = \int_{t_n}^{t_{n+1}} \phi(x) dx \tag{7}$$

And our strategy is to analyze the gap between the desired lifetime distribution in experiment and *real* viewers' residual lifetime distribution in physical world to check whether the testing can be triggered at current time or not.

At current time t, according to the difference value between $Exp(t_0)$ and $Predict(t + t_0)$, there exists two probable cases:

1. $Exp(t_0) \le Predict(t + t_0)$;
2. .. ;

In case (1), when the real value is greater than expected, it is easy to single out $Exp(t_0)$ real viewers for testing to achieve the desired lifetime distribution during $[t + t_0, t + t_1]$.

Then we carry on to inspect next timescale and compute the difference between $Exp(t_1)$ and $Predict(t+t_1)$:

3. $Exp(t_1) \le Predict(t+t_1)$;
4. $Exp(t_1) > Predict(t+t_1)$;

In case (4), when there has no adequate viewers for desired lifetime distribution, we make a further prediction to observe the number of production viewers whose residual lifetime satisfying the condition $\{V \mid V > t_2\}$ at current time t in physical world to confirm whether it is available to perform lifetime control in experiment, and denote the predicted value as $predict(t+t_2)$:

$$Exp(t_1) - Predict(t+t_1) \le predict(t+t_2) \quad (8)$$

If the value of $Exp(t_0)$ is bigger than $Predict(t+t_0)$, just as case (2), we will predict the relationship between the difference value and the number of viewers in physical world with residual lifetime $\{V \mid V > t_1\}$ for orchestration:

$$Exp(t_0) - Predict(t+t_0) \le predict(t+t_1) \quad (9)$$

Only when (9) satisfies, we will move to next timescale $[t+t_1, t+t_2]$. And still, there exists two different possibilities after considering the influence of previous timescale:

5. $Exp(t_1) \le Predict(t+t_1) - (Exp(t_0) - Predict(t+t_0))$;
6. $Exp(t_1) > Predict(t+t_1) - (Exp(t_0) - Predict(t+t_0))$;

In case (6), further evaluation should be made to ensure whether we can achieve the desired lifetime scenario in period of $[t+t_1, t+t_2]$ by orchestrating to let some clients *early-departed* the testing, while case (5) is just what the experiment expected, that is to say:

$$\begin{aligned} &Exp(t_1) - Predict(t+t_1) - \begin{pmatrix} Exp(t_0) - \\ Predict(t+t_0) \end{pmatrix} \\ &\le predict(t+t_2) \\ &\Rightarrow Exp(t_0) + Exp(t_1) \le predict(t+t_0) \end{aligned} \quad (10)$$

In subsequent timescales, all cases possible are included in above analysis considering the influence of previous timescale. Thus if all conditions are satisfied in every timescale until $t_{\exp}$, t can be triggered as t_{start}.

4.2. Independent Arrivals Achieving Global Arrival Pattern

After the triggering condition, orchestrator takes charge of selecting a certain number of real viewers for testing according to the specific lifetime distribution in experiment and adopts an effective mechanism to compute the start times of experiment in each client to create the desired arrival scenario.

Previous research has clearly discussed about how to conduct arrival control with adequate stable viewers in testing channel according to Cox-Lewis Law. But the reality is far more complicated considering the small percentage of stable clients shown in Figure 1. Thus we make a further complement about the distributed algorithm in ShadowStream, and it aims at not only letting clients locally compute its own *virtual* arrival time but also correlating

the specific lifetime distribution with viewers' residual lifetime in physical world. Besides, it will be better if clients with a relatively shorter residual lifetime have a earlier joining time in testing, which reminds us to perform sorting and arrival control based on clients' lifetime.

According to (5), when α is assigned to a specific value, each real viewers can independently compute its average remaining lifetime in physical world, based on the time it has spent in the testing channel at t_{start}. Hence it is not difficult to count the number of clients in each timescale, which denoted as $Real\left(t_{start}+t\right), t \in \left[0, t_{\exp}\right]$. In the process of sorting, it helps to determine the certain number of clients with different residual lifetime by investing the distinction with the desired behavior scenario in current and previous timescale. Specifically, at duration $\left[t, t+\Delta t\right]$ the total number of production viewers required in experiment is not only determined by the specific lifetime scenario, but also includes some additional viewers considering the difference value between the desired and the real lifetime distribution in previous timescale.

When considering the lifetime distribution in current period, let $p\left(t\right)$ be the ratio of real viewers required in experiment to the total number of available real viewers in physical world. If $p\left(t\right)<1$, which indicates adequate real viewers for the specific experimental lifetime distribution and lifetime control in current period, each client with $\left\{V \mid \mathrm{t}<V<t+\Delta t\right\}$ will independently participate in the scenario with probability $p\left(t\right)$, and compute its *virtual* arrival time in experiment. And this leads to a simple distributed algorithm shown in Table 1.

4.3. Lifetime Control

In theory, lifetime control in ShadowStream should be more intricate when compared with arrival control due to the use of real viewers, whose behavior in physical world seems to have a greater influence on the actual lifetime distribution in experiment. However, in arrival control, we have managed to sort out testing clients in accordance with the desired lifetime distribution, which, on the other hand, has greatly relieved the pressure in the process of lifetime control. More specifically, the purpose of lifetime control is to make a certain number of clients whose average residual lifetime V satisfying the condition $\left\{V \mid V>t+\Delta t\right\}$ virtually depart the channel during $\left[t, t+\Delta t\right]$, when there exists inadequate testing clients for the specific lifetime distribution in experiment, and $t \in \left[0, t_{\exp}\right]$.

To make clients' lifetime in experiment matches the real scenario as much as possible, we adopt the principle of proximity to pick out testing clients to perform lifetime control. When testing viewers' residual lifetime cannot satisfy the desired distribution in experiment, orchestrator select a certain number of clients that have the minimum residual lifetime difference with $\left(t+\Delta t\right)$ from a global view, based on the UDP report from clients which includes its average residual lifetime V_i. We define the maximum residual lifetime as V_m among those clients as an index in lifetime control. Meanwhile, the orchestrator computes $q\left(t\right)$ as the ratio of the difference between the desired and the real lifetime distribution in experiment to the total number of testing clients with $\left\{V \mid V \le V_m\right\}$. The process of lifetime control is introduced in Table 2.

Unlike the process of arrival control, we don't apply distributed algorithm to let testing clients compute its own lifetime in experiment once they join the testing, but choose to perform lifetime control when necessary. Thus in the process of lifetime control, we just manage to deal with the case by shortening a certain number of clients' lifetime in experiment.

Table 1. Algorithm combined with sorting and decentralized control for each client i to compute arrival time $a_{e,i}$

Client i at t_{start} :

02. Compute its mean residual lifetime V_i according to $V_i = \frac{t}{\alpha - 1}$

Orchestrator:

02. Let $Real\left(t_{start} + t\right)$ be the total number of available clients with $\left\{V \mid t < V \le t + \Delta t\right\}$ in physical world

03. Let $Exp\left(t\right) = \int_{t}^{t+\Delta t} \phi\left(x\right) dx$

04. Let $dif\left(t\right)$ be the difference value between the desired and the real lifetime duration in physical world until time t

05. Let $p\left(t\right) = \frac{Exp\left(t\right) + dif\left(t\right)}{Real\left(t_{start} + t\right)}$

Check : **if** $p\left(t\right) > 1$,

06. Send t_{start} , $t_{\exp}$ and $\lambda\left(t\right)$ to each client with $\left\{V \mid t < V \le t + \Delta t\right\}$

07. Set $dif\left(t + \Delta t\right) = Exp\left(t\right) + dif\left(t\right) - Real\left(t_{start} + t\right)$

Client i , upon receiving t_{start} , $t_{\exp}$ and $\lambda\left(t\right)$:

08. Draw waiting time ω_i according to $F\left(t\right) = \frac{\Lambda\left(t\right)}{\Lambda\left(t_{\exp}\right)}$

09. Compute arrival time: $a_{e,i} = t_{start} + \omega_i$

else,

06. Send t_{start} , $t_{\exp}$, $\lambda\left(t\right)$ and $p\left(t\right)$ to each client with $\left\{V \mid t < V \le t + \Delta t\right\}$

07. Set $dif\left(t + \Delta t\right) = 0$

Client i , upon receiving t_{start} , $t_{\exp}$, $\lambda\left(t\right)$ and $p\left(t\right)$:

08. **if** $random\left(\ \right) > p\left(t\right)$ **then return**

09. Draw waiting time ω_i according to $F\left(t\right) = \frac{\Lambda\left(t\right)}{\Lambda\left(t_{\exp}\right)}$

10. Compute arrival time : $a_{e,i} = t_{start} + \omega_i$

Table 2. Algorithm for those clients with $\{V \mid V < V_m\}$ to perform lifetime control and compute residual lifetime V_i.

Orchestrator:

01: Let $Real'(t_{start}+t)$ be the total number of available clients with $\{V \mid t < V \le t+\Delta t\}$ in experiment

02. **if** $Real'(t_{start}+t) \ge Exp(t)$ **then return**

03. Let $dis(t) = Exp(t) - Real'(t_{start}+t)$

04. Assign the value of V_m according to $dis(t)$

05. Let Sum be the total number of clients with $\{V \mid V \le V_m\}$

Client i $(V_i \le V_m)$, upon receiving t_{start}, $t_{\exp}$, $q(t)$ and $\phi(t)$:

08. **if** $random(\ ) > q(t)$ **then return**

09. Draw waiting time θ_i according to $F'(t) = \dfrac{\Phi(t)}{\Phi(t_{\exp})}$

10. Compute residual lifetime: $V_i = t_{start} + \theta_i$ and refresh it

4.4. Stable Clients' Rejoining

Our previous work is all based on an idea that each client just participates in experiment once. But in this section, we intend to take full advantages of stable viewers in physical world to let them rejoin the testing following the expected behavior pattern once they have virtually departed the experiment to further increase real viewers' utilization level.

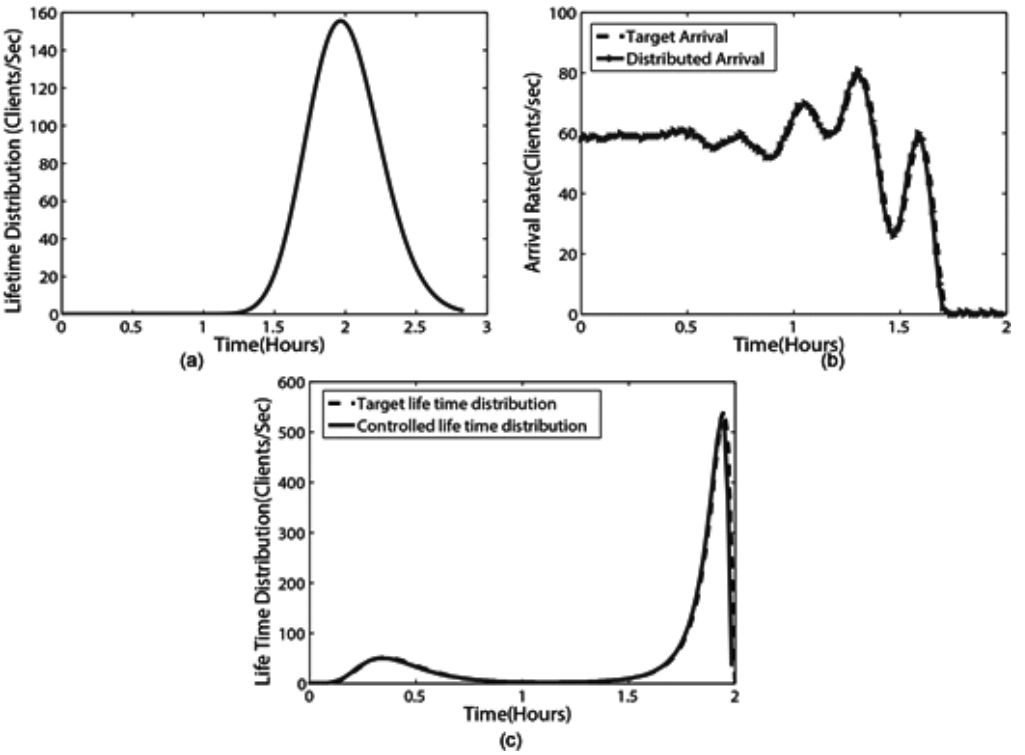
To describe the status of a production viewer, we introduce a *Stability Index* ($s-Index$) (Wang&Liu, 2008) here to characterize its degree of stability, denoted as SI. Once the threshold of SI_{thr} is defined from a global perspective, the number of stable clients M ($SI \ge SI_{thr}$) that are available for rejoining in physical world can also be determined.

When detecting an *early-departed* stable client in experiment, orchestrator will immediately perform distributed arrival algorithm according to Cox-Lewis law and let it compute its own arrival time for next testing. But on the other hand, this method is largely confined to the number of stable clients in physical world and the specific behavior scenario we assigned. To integrate the process of rejoining in experiment, there are two key points concerned since orchestration is executed based on lifetime: 1) the time when the first stable testing client joins; 2) the time when the last unstable testing client departs in testing. Those two event times can be computed according to the desired arrival rate function $\lambda(t)$ and the specific lifetime distribution $\phi(t)$, denoted as t_a, t_d respectively. And at any time t, the value of stable clients M in experiment should satisfy the following conditions:

$$\int_{t_a}^{t} \lambda(x)dx < M, \forall t \in [t_a, t_d] \tag{11}$$

$$\int_{t_d}^{t} \lambda(x)dx - \int_{t_d}^{t} \phi(x)dx < M, \forall t \in [t_d, t_{\exp}] \tag{12}$$

Figure 7. The gap between the expected and the actual behavior scenario by performing arrival/lifetime control. (a) Lifetime distribution in physical world. (b) Difference between the target arrival pattern and the real by performing distributed control. (c) Difference between the target lifetime distribution and the real by performing lifetime control.



5. EVALUATIONS

In this section, we conduct an evaluation of experiment orchestration on MATLAB to testify whether we can attain the expected behavior pattern. Firstly, we evaluate the situation when taking the process of matching in experiment orchestration. Then we use relatively few but stable viewers for testing and attempt to make them rejoin the experiment to assess the performance of the orchestrator. At last, we analyze the utilization level of real viewers by applying different algorithms.

5.1. Setting

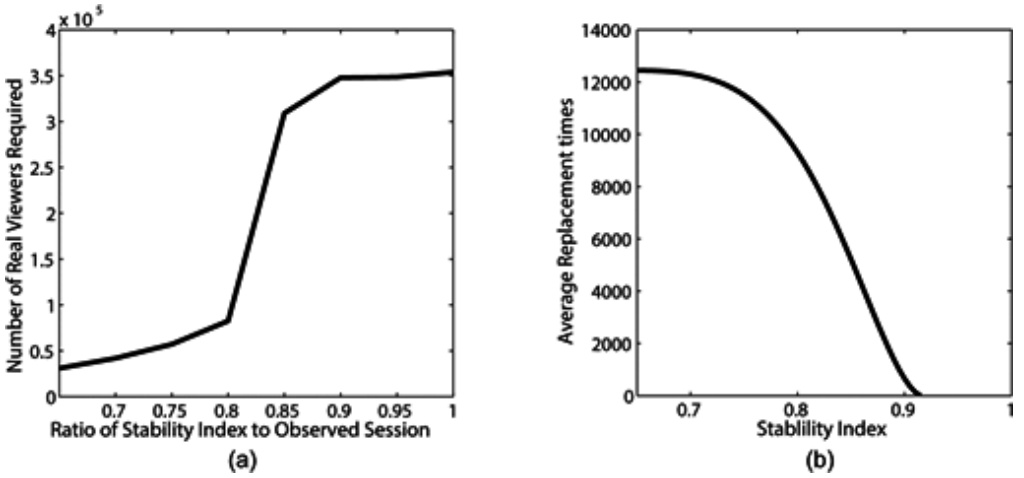
To evaluate the experiment orchestration under the specific lifetime distribution in physical world, we assume that real viewers' residual lifetime obeys the Chi-square distribution with 120-freedom shown in Figure 7(a) according to (5) when $\alpha = 2$ (for simplicity), with the testing duration $t_{\exp} = 2\left(Hour\right)$. Besides we define a target arrival rate function shown in Figure 7(b), which has 348,618 arrivals when instantiated in ShadowStream, and a lifetime distribution in Figure 7(c),

5.2. Evaluation with Matching

Here we make an evaluation of the experiment orchestrator by matching each client's residual lifetime in physical world with the specific behavior pattern.

And the difference between the target and the actual arrival/lifetime scenario resulting

Figure 8. Trade-off between the minimum real viewers required and average replacement times in the process of stable clients' rejoining. (a) Minimum number of real viewers required under different value of SI_{thr}. (b) Average number of replacement times under different value of SI_{thr}.



from orchestration in experiment is presented in Figure 7(b) and Figure 7(c) respectively. We can see that it is possible to fill the gap by the process of matching, which means there is no *early-quitted* client in experiment.

It should be noted that in real live-streaming networks, predictions about real viewers' behavior in the near future have no way to be 100% accurate considering that a client may depart due to insufficient streaming quality or some uncertain factors, which is hard to direct model. Thus, in a live-testing streaming network, replacement is inevitable even the evaluation outcome shows no necessary for it.

5.3. Evaluation with Stable Clients' Rejoining

To better understand the trade-off in the process of rejoin, we present the minimum number of real viewers required in physical world and the average rejoining times for testing clients under different value of SI_{thr} in Figure 8. And it apparently shows that the total number of real viewers required in physical world increases as the increment of SI_{thr}, while the average replacement times in experiment decreases.

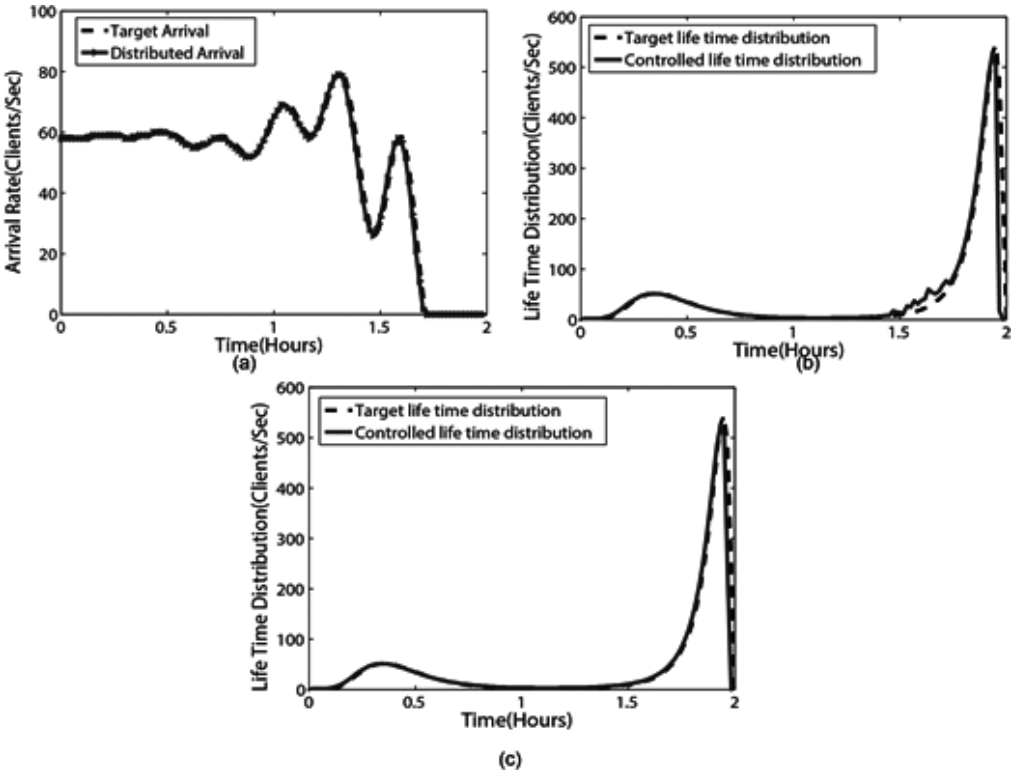
At first, we define $SI_{thr} = 0.67$, which means only those clients whose $SI_{thr} \geq 0.67$ are stable enough for rejoining in experiment. The outcome is shown in Figure 9(a) and Figure 9(b). And it is clear to see that, gap between the target and the actual arrival scenario is small enough to be negligible, while there exists significant difference between the desired and the actual lifetime distribution in experiment. Thus in this situation, it requires some more real viewers in physical world to replace those *early-quitted* clients in the process of experiment.

Then we redefine $SI_{thr} = 1$ and present the actual lifetime distribution in experiment in Figure 9(c). And it shows that the gap is closed when comparing with Figure 9(b), which indicates that the average replacement times in experiment is significantly decreased.

5.4. Performance Comparison

Having observed the gap between the desired and the real behavior scenario to inspect the probable replacement times in experiment, we make a further evaluation to observe real viewers' utilization level in physical world, which

Figure 9. The gap between the expected and the actual behavior scenario by letting stable clients in experiment rejoin the testing with a new identity. (a) Difference between the target arrival pattern and the real by making stable clients rejoin with $SI_{thr} = 0.67 \bullet t_{\exp}$. (b)Difference between the target lifetime distribution and the real by making stable clients rejoin with $SI_{thr} = 0.67 \bullet t_{\exp}$. (c) Difference between the target lifetime distribution and the real by making stable clients rejoin with $SI_{thr} = t_{\exp}$.



will affect the scale of experiment to a large degree. Since the behavior scenario is specified, we use the minimum value of real viewers $M_{\min}$ required in physical world whose real residual lifetime satisfies the distribution illustrated in Figure 9(a) to describe the utilization level in experiment, and the result is shown in Table 3, which manifests when considering matching in experiment the required number of viewers in physical world has nearly reduced by half compared with our previous research. By making stable clients rejoin the testing, real viewers' utilization level is further improved based on the value of SI_{thr}. Specifically, in average replacement times equal circumstances, $M_{\min}$ has been decreased from 360,018 to 353,700 after taking full advantages of stable clients in experiment.

By making a comparison among the results including the average replacement times in experiment as well as the minimum value of real viewers required in physical world, it is not hard to draw an conclusion that by performing matching and rejoining in the process of orchestration, real viewers' utilization level is obviously improved when minimum average replacement times is ensured.

Table 3. Clients' utilization level

Scenario	Minimum Value
Orchestration without matching or rejoining ($SI_{thr} = 1$)	$M_{\min} = 698,238$
Orchestration with matching	$M_{\min} = 360,018$
Orchestration with matching and rejoining ($SI_{thr} = 0.67$)	$M_{\min} = 34,135$
Orchestration with matching and rejoining ($SI_{thr} = 1$)	$M_{\min} = 353,760$

6. CONCLUSION

The limitation of the original ShadowStream system is that only stable streaming viewers can be selected for entering the live testing. This significantly limits the scale of tests. In this paper, a novel distributed client lifecycle control method is developed. By matching the desired scenario with real viewers' behaviors, we demonstrate that the scale of experiments can be doubled.

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their valuable comments. This work is partially supported by "National Natural Science Foundation of China (No. 61202107, No. 61100220, No. 61202303)", by "National High Technology Research and Development Program of China (863 Program No. 2014AA01A702)" and by the "Fundamental Research Funds for the Central Universities". Chen Tian is the corresponding author.

REFERENCES

Banerjee, S., Bhattacharjee, B., & Kommareddy, C. (2002). Scalable application layer multicast: Vol. 32. *No. 4* (pp. 205–217). ACM.

Bavier, A., Feamster, N., Huang, M., Peterson, L., & Rexford, J. (2006, September). In VINI veritas: Realistic and controlled network experimentation. []. ACM.]. *Computer Communication Review*, *36*(4), 3–14. doi:10.1145/1151659.1159916

Bishop, M. A., Rao, S. G., & Sripanidkulchai, K. (2006, April). *Considering Priority in Overlay Multicast Protocols Under Heterogeneous Environments* (pp. 1–13). INFOCOM. doi:10.1109/INFOCOM.2006.140

Bonald, T., Massoulié, L., Mathieu, F., Perino, D., & Twigg, A. (2008, June). Epidemic live streaming: Optimal performance trade-offs. []. ACM.]. *Performance Evaluation Review*, *36*(1), 325–336. doi:10.1145/1384529.1375494

Castro, M., Druschel, P., Kermarrec, A. M., Nandi, A., Rowstron, A., & Singh, A. (2003, October). SplitStream: High-bandwidth multicast in cooperative environments. []. ACM.]. *Operating Systems Review*, *37*(5), 298–313. doi:10.1145/1165389.945474

Chang, H., Jamin, S., & Wang, W. (2009, November). Live streaming performance of the Zattoo network. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference* (pp. 417-429). ACM. doi:10.1145/1644893.1644944

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., & Bowman, M. (2003). Planetlab: An overlay testbed for broad-coverage services. *Computer Communication Review*, *33*(3), 3–12. doi:10.1145/956993.956995

Cisco, I. (2012). Cisco visual networking index: Forecast and methodology, 2011--2016. *CISCO White paper*, 2011-2016.

Cox, D. R., & Lewis, P. A. (1966). The statistical analysis of series of events.Monographs on Applied Probability and Statistics, London: Chapman and Hall, 1966, 1.

Dischinger, M., Haeberlen, A., Beschastnikh, I., Gummadi, K. P., & Saroiu, S. (2008, August). Satellitelab: Adding heterogeneity to planetary-scale network testbeds. []. ACM.]. *Computer Communication Review*, *38*(4), 315–326. doi:10.1145/1402946.1402994

Kim, M. S., Kim, T., Shin, Y., Lam, S. S., & Powers, E. J. (2004). A wavelet-based approach to detect shared congestion: Vol. 34. *No. 4* (pp. 293–306). ACM.

Li, B., Xie, S., Qu, Y., Keung, G. Y., Lin, C., Liu, J., & Zhang, X. (2008, April). Inside the new coolstreaming: Principles, measurements and performance implications. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE.

Magharei, N., & Rejaie, R. (2009). Prime: Peer-to-peer receiver-driven mesh-based streaming. [TON]. *IEEE/ACM Transactions on Networking*, *17*(4), 1052–1065. doi:10.1109/TNET.2008.2007434

Picconi, F., & Massoulié, L. (2008, September). Is there a future for mesh-based live video streaming? In *Peer-to-Peer Computing, 2008. P2P'08. Eighth International Conference on* (pp. 289-298). IEEE. doi:10.1109/P2P.2008.18

Sundaresan, S., De Donato, W., Feamster, N., Teixeira, R., Crawford, S., & Pescapè, A. (2011, August). Broadband internet performance: A view from the gateway. []. ACM.]. *Computer Communication Review*, *41*(4), 134–145. doi:10.1145/2043164.2018452

Swartz, G. B. (1973). The mean residual lifetime function. *Reliability*. *IEEE Transactions on*, *22*(2), 108–109.

Tian, C., Alimi, R., Yang, Y. R., & Zhang, D. (2012, August). ShadowStream: performance evaluation as a capability in production internet live streaming networks. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication* (pp. 347-358). ACM. doi:10.1145/2342356.2342430

Wang, F., Liu, J., & Xiong, Y. (2008, April). Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE.

White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., ... & Joglekar, A. (2002). An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review, 36*(SI), 255-270.

White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., ... & Joglekar, A. (2002). An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review, 36*(SI), 255-270.

Wu, C., Li, B., & Zhao, S. (2008, April). Multi-channel live p2p streaming: Refocusing on servers. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE.

Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., . . . Li, B. (2009, October). Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In *Proceedings of the 17th ACM international conference on Multimedia* (pp. 25-34). ACM.

Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., . . . Li, B. (2009, October). Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In *Proceedings of the 17th ACM international conference on Multimedia* (pp. 25-34). ACM.

Zhou, Y., Chiu, D. M., & Lui, J. C. (2007, October). A simple model for analyzing P2P streaming protocols. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on* (pp. 226-235). IEEE. doi:10.1109/ICNP.2007.4375853