Cooperative On-Demand Delivery for IPTV Networks

Aytac Azgin, Ghassan AlRegib, and Yucel Altunbasak
School of Electrical and Computer Engineering
Georgia Institute of Technology

***Abstract*—In this paper, we investigate the use of cooperative transmission strategies to support timely and efficient delivery of on-demand content to end users. Within the proposed framework, a cooperative transmission strategy suggests that users who have access to the requested content cooperatively transmit to the targeted user(s) to minimize the servicing requirements at the server side and to improve the scalability performance in the network. Through extensive simulation-based studies, we show that significant performance improvements can be achieved with the proposed framework to enable efficient access to an ever growing on-demand content.**

I. Introduction

In recent years, user demand for the delivery of broadcast content over IP infrastructure has seen a significant increase [1]. Furthermore, through rapidly evolving server-assisted on-demand content delivery services, live broadcast services, and peer-assisted on-demand streaming services, we have observed a substantial increase in the amount of content that is immediately made available to the end users [2–4]. Due to this mutually evolving content delivery cycle, the amount of bandwidth required to timely service the user requests has also seen a significant increase [5, 6]. To continue to support the diverse service quality requirements for the continually increasing user demand, it is of critical importance to come up with resource efficient content delivery techniques.

For on-demand delivery systems, a substantial portion of the network load can be attributed to the most popular content. That is because, as the popularity of the content increases linearly, the number of users receiving such content increases exponentially [7–10]. To overcome the sudden increase in user demand, caching policies are typically utilized to distribute the load in the network by giving higher priority to the delivery of more popular content. However, as the number of users increase beyond a certain threshold, even the distributed caching policies can become inefficient to handle the requested load. Considering the bandwidth requirements for the received requests and the activity duration associated with the content requested, it becomes necessary to further distribute the network load to more efficiently manage the available resources.

To achieve these objectives, we propose a cooperative-diversity driven content delivery framework that utilizes session peers to efficiently distribute the network resources. Specifically, in the proposed framework, the users that have immediate access to the requested content transmit together with similar users to maximize the resource usage efficiency in the network. In wireless networks, cooperative diversity is essentially utilized to minimize the energy consumption in the network, and, in doing so, maximize the network lifetime [11]. However, because of the oftentimes ad-hoc nature of the network, wireless cooperation is typically limited to the use of individual nodes, and as a result, performance improvements strongly depend on the network topology. Wireline networks, on the other hand, do not possess such limitations, because of the typical use of dedicated servers to support the content delivery process, thereby offering significant advantages to accommodate the system needs. In short, in our study, we apply the cooperation principle on wireline-based on-demand content delivery systems to achieve the following objectives: (i) maximize the servicing capacity in the network by minimizing the servicing overhead through the dedicated server, and in doing so, significantly improve the scalability performance of the network, and (ii) increase the operational lifetime of the network by distributing resources to session peers as fairly as possible. [1]

To achieve our objectives, we propose the Cooperative Weighted Fair Queueing (C-WFQ) technique, which fairly and efficiently allocates the peer resources to the ongoing sessions. The analysis of the proposed framework shows the significant advantages of utilizing cooperative diversity principle in on-demand content delivery networks.

The rest of the paper is organized as follows. In Section II we present our system model. We discuss the algorithmic details of the C-WFQ technique in Section III. We analyze the performance of the proposed cooperative delivery framework in Section IV. Section V concludes our paper.

II. System Model

Assume that there are $k+1$ users that request session s_j's multicast within a certain timeframe, which we refer to as ΔT and which initiates with a user request. The request times are given by $T_{req,ij}$, where $1 \leq i \leq k+1$ and $(T_{req,ij} - T_{req,1j}) \leq \Delta T$, $\forall i \in s_j$. Ideally, each user is expected to join the targeted session's multicast at the time of request. However, since the on-demand content is typically delivered as a whole, starting from the beginning, it becomes impractical to initiate a session for each received request, especially when the sessions are long and the requests arrive on a frequent basis.

To circumvent these limitations, we propose the cooperative delivery framework shown in Figure 1, where, after a client makes a session-join request to the dedicated server, it also joins the source multicast. [2] To guarantee an almost instant playout, earlier transmitted packets (through the session multicast) are delivered to the client at a rate that at least equals the source multicast rate. [3] To deliver these packets, the proposed architecture uses the session peers. Specifically, each user that has partial/complete access to the requested content becomes a potential source for the cooperative delivery process. In short, for all received requests, two delivery sources exist, *i.e.*, session peers and the dedicated server. However, dedicated server is only used to compensate for

[1]Here, the term operational lifetime may refer to measures such as the willingness or capability of users to cooperate. For instance, if each user has a limited uplink budget (*e.g.*, X bytes per month), then unfair resource allocation may cause some users to consume their budget earlier, thereby, limiting cooperation gains.

[2]In Figure 1 Req_i represents the session join request made by ν_i, $S_{data_{S\to i}}$ represents the server data delivered to user ν_i, and $P_{data_{i\to\{j,k\}}}$ represents the peer data delivered from ν_i to ν_j and ν_k.

[3]In our framework, clients are assumed to have a downlink capacity of $> 2 \times W_M$, where W_M represents the source multicast rate.

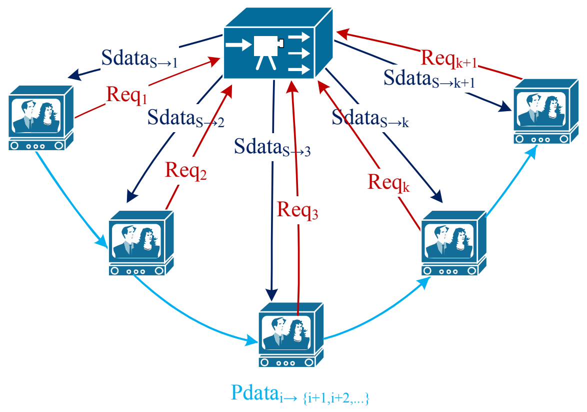


Fig. 1. Cooperative transmission framework.

the lack of resources at the user side, when the session peers cannot provide the sufficient resources required to achieve the minimum delivery requirements for the given request. In our framework, we define the *minimum delivery requirements* as follows:

- The latency for decoding the requested on-demand content should be equal to (if not less than) the latency associated with unicast delivery. [4]

An important parameter of choice for the proposed architecture is the ΔT parameter, which represents the minimum waiting time for the content delivery server to create a new multicast session. As the multicast session length increases, it becomes crucial to establish breathing instances in time, where the system generates a new multicast session to accommodate for the needs of the newly arriving requests without violating the system constraints. If the ΔT value is not properly assigned, clients may experience huge, and oftentimes unacceptable, session initialization delays. Choosing a small/large value for ΔT determines how efficiently system resources are allocated among session peers and the dedicated server. For instance, by choosing a large value for ΔT, we would be increasing the burden on the session peers, whereas, by choosing a small value for ΔT, we would be increasing the burden on the dedicated server. In short, to optimize the resource usage efficiency in the network, we need to carefully study the relationship between request arrival process and ΔT value, and evaluate the associated tradeoffs.

We next give an in-depth overview and analysis of the proposed cooperative content delivery technique that aims to improve the resource usage efficiency in the network.

III. Cooperative Weighted Fair Queueing (C-WFQ) Approach

Cooperative Weighted Fair Queuing (C-WFQ) approach is a dynamic resource allocation technique that focuses on the individual transmission subperiods to assign peer resources to the received requests. [5] As its name suggests, C-WFQ is a weight-based resource allocation technique, which uses the dynamically assigned user weights to determine the session peers' contributions during a cooperative delivery phase.

C-WFQ initializes by assigning unit weight to each user that has newly joined the currently active session. Then, at the beginning of each transmission subperiod, it recalculates the user weights based on the earlier peer contributions, for the given session. Here, since a session peer's contribution is not limited to a specific timeframe, after each request, we probe all the available peers to request their support. Additionally, each session peer is allowed to service multiple requests at once, as in the case of Weighted Fair Queuing, by properly distributing the available resources to all the active requests.

We illustrate the operation of the C-WFQ approach in Figure 2 (from the user ν_l's perspective), where we assume the following: processing of all requests received before the $(k-1)$th request finishes by the time $(k-1)$th request is made. At each new request arrival or service completion event, available resources at ν_l are reallocated to the active requests. We next explain in detail the operation of the proposed delivery framework.

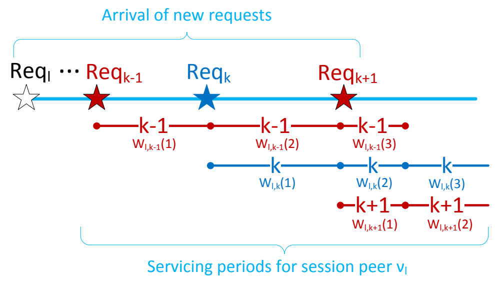


Fig. 2. Transmission events for the C-WFQ approach.

Assume that user ν_i makes a request for session s_j at time $T_{i,j}$. For each session, the counter that represents the user count starts at 1, hence, by $T_{i,j}$, the number of additional requests that arrive for s_j is equal to $i-1$. For the time being, let us also assume that users have unlimited bandwidth capacity along the downlink channel, allowing any number of session peers to be utilized during the content delivery phase. We can therefore use the following equation to find the weights for peer contributions:

$$\omega_k(T_{i,j}) = \left(\left\lceil \frac{B_{j,\max} - B_k(T_{i,j})}{B_{j,\max}} \right\rceil^+\right)^{\beta_{i,k}} \tag{1}$$

where, for $k < i$, $\omega_k(T_{i,j})$ represents the weight assigned to user ν_k at time $T_{i,j}$, $B_{j,\max}$ represents the maximum amount of data each session peer is allowed to deliver during the lifetime of s_j's multicast, $B_k(T_{i,j})$ represents the contribution from ν_k by $T_{i,j}$, and $\beta_{i,k} > 1$ represents the normalization metric assigned to ν_k for ν_i's request. [6] Therefore, by choosing a $\beta_{i,k}$ value that is greater then 1, the more a session peer contributes, the faster its weight decreases, eventually converging to 0.

After ν_i makes a join request to receive session s_j at time $T_{i,j}$, we can determine the size of expected delivery from each session peer by using the following equation:

$$D_{k,i} = \frac{w_k(T_{i,j}) \times W_M \times (T_{i,j} - T_{1,j})}{\sum_{l=1}^{i-1} w_l(T_{i,j})} \tag{2}$$

where, for $k < i$, $D_{k,i}$ represents the amount of data that ν_k is expected to deliver in response to ν_i's request. Note that, since client ν_i joins the session multicast immediately, session peers are only expected to deliver the data that corresponds to the timeframe $[T_{1,j}, T_{i,j}]$. [7]

[4]Specifically, user perceived decoding latency performance should assume as if the content is delivered through a unicast streaming server dedicated to servicing each user's request independently.

[5]Here, a transmission subperiod initiates after a new request is received or the servicing of an earlier request finishes

[6]$\lceil x \rceil^+ = \max(x, 0)$.

[7]Note that, in practice, the delivery of the session multicast can at the earliest start at time $\left(T_{i,j} + T_J\right)$, where T_J represents the session join latency. For the sake of simplicity, and since its value is much less than the duration of a typical delivery timeframe (*e.g.*, 0.1*s* to tens of seconds), we omitted its impact in our calculations.

We next determine how to effectively distribute the available peer resources to multiple requests' use. Since C-WFQ allows session peers to target multiple clients at once, if resource availability at a session peer is not sufficient to deliver the requested content at the desired rate, then a resource reallocation policy is needed to make the best use of the available resources at the given peer. For that purpose, we use the following methodology.

We initiate the resource reallocation phase by finding the minimum required delivery rate for each session peer (at the time of user ν_i's session join request) as follows:

$$W_{kl,\min}(T_{i,j}) = \frac{D_{k,l}(T_{i,j})}{(T_{l1,j} - T_{il,j})} \tag{3}$$

where for $k < l$ and $i > l$, $W_{kl,\min}(T_{i,j})$ represents the minimum required delivery rate for session peer ν_k in response to ν_l's request at time $T_{i,j}$, $D_{k,l}(T_{i,j})$ represents the updated value for the expected delivery from ν_k to ν_l at $T_{i,j}$, and $T_{il,j}$ equals $\big(T_{i,j} - T_{l,j}\big)$. [8]

Next we determine the total amount of resources that ν_k needs to allocate for the active requests it has received at the time ν_i makes its join request:

$$W'_{k,\min}(T_{i,j}) = \sum_{l=k+1}^{i} W_{kl,\min}(T_{i,j}) \tag{4}$$

where $W'_{k,\min}(T_{i,j})$ represents the minimum required delivery rate by ν_k at $T_{i,j}$.

We can then calculate the initial rate assigned by ν_k to each active request it has received as follows:

$$W_{k,l}(T_{i,j}) = \begin{cases} \dfrac{\big(W_u^{(k)} - W'_{k,\min}(T_{i,j})\big)}{\sum_{\forall m}\big(\zeta^*_{k,m}/\zeta^*_{k,l}\big)} & \\ +W_{kl,\min}(T_{i,j}) & \text{if } \dfrac{W'_{k,\min}(T_{i,j})}{W_u^{(k)}} < 1 \\ \dfrac{W_u^{(k)}}{\sum_{\forall m}\big(\zeta^*_{k,m}/\zeta^*_{k,l}\big)} & \text{otherwise} \end{cases} \tag{5}$$

where $W_u^{(k)}$ represents the available uplink capacity at user ν_k, $\zeta^*_{k,l}$ equals $\big(\gamma_{k,l} \times W_{kl,\min}(T_{i,j})\big)$, and $\gamma_{k,l}$ represents the weight assigned to ν_l's request at ν_k. Here, weight parameters are mainly used to prioritize certain requests, for instance, requests with shorter deadlines (*i.e.*, earlier received requests) or higher resource needs (*i.e.*, latest received requests). The default value for $\gamma_{k,l}$ is equal to 1.

We next update the initially assigned weights based on the overall resource availability for each active request to further reduce the server contribution. Specifically, assume that, for ν_l's request, the amount of available resources at the session peers is given by:

$$\Upsilon_l(T_{i,j}) = \sum_{\forall k \in C(l)} \widehat{W}_{kl,\min}(T_{i,j}) \tag{6}$$

where $C(l)$ represents the cooperation set for ν_l's request and $\widehat{W}_{kl,\min}(T_{i,j})$ is determined as follows:

$$\widehat{W}_{kl,\min}(T_{i,j}) = W_u^{(k)} \times \frac{W_{kl,\min}(T_{i,j})}{\sum_{\forall m} W_{km,\min}(T_{i,j})} \tag{7}$$

[8]Note that, a similar process is used to update the delivery rates after the servicing of a request finishes.

Assume that $W_{M,l}(T_{i,j})$ represents the minimum delivery rate requirement for ν_l's request at time $T_{i,j}$ [9]. Since we aim to minimize the resource usage rate at the server side, we first determine whether $\Upsilon_l(T_{i,j})$ is less than or greater than $W_{M,l}(T_{i,j})$ and by how much its value differs from $W_{M,l}(T_{i,j})$ (*i.e.*, $\gamma_l(T_{i,j}) = |\Upsilon_l(T_{i,j}) - W_{M,l}(T_{i,j})|$). If, for instance, $\Upsilon_l(T_{i,j}) > W_{M,l}(T_{i,j})$, then we can allocate a portion of these resources, which amounts to $\gamma_l(T_{i,j})$ bits per second, to other requests without violating the service quality requirements for ν_l's request. Assume that Γ^+ represents the set of requests, for which the available resources at the session peers are more than sufficient to satisfy the service quality requirements (*i.e.*, $\forall l \in \Gamma^+$, $\Upsilon_l(T_{i,j}) > W_{M,l}(T_{i,j})$), and similarly, Γ^- represents the set of requests, for which the available resources at the session peers are not sufficient to satisfy the service quality requirements (*i.e.*, $\forall l \in \Gamma^-$, $\Upsilon_l(T_{i,j}) \leq W_{M,l}(T_{i,j})$). We can then determine the total amount of resources that can be reallocated to other requests as follows:

$$\Upsilon^+ = \sum_{\forall l \in \Gamma^+} \big(\Upsilon_l(T_{i,j}) - W_{M,l}(T_{i,j})\big) \tag{8}$$

Since the server is used to support the delivery process only if the session peers cannot deliver at a rate that would ensure the timely delivery of the requested content, to minimize the amount of resources delivered through the dedicated server, we need to ensure that, at any given point in time, each request can be serviced at least at the minimum required rate. Hence, we can determine the additional resources that would initially be requested from the server at $T_{i,j}$ as follows:

$$\Upsilon^- = \sum_{\forall l \in \Gamma^-} \big\lceil W_{M,l}(T_{i,j}) - \Upsilon_l(T_{i,j})\big\rceil^+ \tag{9}$$

Note that, if $\Upsilon^+ > \Upsilon^-$, then resource requirements for all the available requests can be met without requesting additional assistance from the dedicated server. [10] In short, resources at the session peers are reallocated as follows:

$$W^*_{k,l}(T_{i,j}) = \begin{cases} \widehat{W}_{kl,\min}(T_{i,j}) \times \left[1 - \dfrac{\min(\Upsilon^-, \Upsilon^+)}{\Upsilon^+} \right. & \\ \left. \times \Big(1 - \dfrac{W_{M,l}(T_{i,j})}{\Upsilon_l(T_{i,j})}\Big)\right] & \text{if } l \in \Gamma^+ \\ \widehat{W}_{kl,\min}(T_{i,j}) \times \dfrac{W_{M,l}(T_{i,j})}{\Upsilon_l(T_{i,j})} & \text{if } l \in \Gamma^- \end{cases} \tag{10}$$

After the resources are reallocated at the session peers, we also need to make sure that the local bandwidth constraints at the session peers can be met. For instance, if at ν_k, $\sum_{\forall l} W^*_{k,l}(T_{i,j}) > W_{u,k}$, then we need to reduce the delivery rates assigned by ν_k to each of its active requests. In a similar way, if at ν_k, $\sum_{\forall l} W^*_{k,l}(T_{i,j}) < W_{u,k}$, then it becomes possible to assign more resources to the active requests at ν_k. For that purpose, we use a recursive procedure that reallocates the resources at the session peers until no session remains with additional resources requested from the server side, or no additional resources can be delivered from the peer side.

Finally, allocated resources at the session peers need to be updated whenever the servicing of a request finishes and the

[9]At the limit, when the resources at the session peers are barely enough to service all the active requests, $W_{M,l}(T_{i,j})$ is expected to be equal to W_M.

[10]The given statement is valid as long as the session peers have full access to the requested resources.

TABLE I
SIMULATION PARAMETERS

L_S	100 $min.$
ΔT	$5 \times \kappa$ $min.$, where $\kappa \leq L_S/5$
W_d	15 $Mbps$
W_u	1 $Mbps$
W_M	3 $Mbps$
λ_S	1/20 $reqs/sec$
$B_{\min}$	$2 \times L_S \times W_M$ $bits$
$B_{\max}$	$2 \times \big(W_u/W_M\big) \times B_{\min}$ $bits$

resources used by the given request are released at the session peers. For that purpose, a similar procedure is used to redistribute the released resources to the active requests starting with the request that receives resources through the dedicated server at the highest rate.

IV. PERFORMANCE ANALYSIS

In this section, using a simulation-based study, we investigate the performance of the proposed content delivery framework when the request arrival process is modeled by using the (homogeneous) Poisson process. We list the simulation parameters in Table I. Here, L_S represents the session length, ΔT represents the average time spacing between two successive sessions, W_u (or W_d) represents the average servicing capacity along the users' uplink (or downlink) channel [11], W_M represents the delivery rate for the source multicast, λ_S represents the mean arrival rate for the session join requests, and $B_{\min}$ represents the maximum amount of data that can be delivered (at the source multicast rate) during the activity lifetime of a session. [12] The value of $B_{\max}$ is chosen to ensure that client ν_i that transmits continuously during the lifetime of session s_j ends up with a weight of $w_i(T_{1,j} + 2L_{S,j}) \geq 0$ by the end of s_j's activity lifetime.

In our study, we mostly focus on the following performance measures: average and maximum servicing overhead, synchronization latency, and fairness index for peer-bandwidth usage. In our framework, synchronization latency refers to the delivery time of unicast-transmitted session data, and we use fairness index to measure how resources are distributed to session peers. To obtain the fairness results we use Jain's fairness index, which is defined based on the following equation [12]:

$$F_{\nu,j} = \frac{\left(\sum_{i=1}^{N_j} B_i(\Delta T_j)\right)^2}{N_j \times \sum_{i=1}^{N_j} B_i(\Delta T_j)^2} \tag{11}$$

where $F_{\nu,j}$ represents the level of fairness observed in bandwidth usage for peers connected to session s_j, and N_j represents the number of peers connected to session s_j during the same period (*i.e.*, Δ_T). The value of $F_{\nu,j}$ ranges from $1/N_j$ (worst-case scenario) to 1 (best-case scenario). [13]

We first analyze the overhead performance at the dedicated server. In Figure 3 we show the results corresponding to the average servicing overhead, $E[W_S]$, as we vary the value of ΔT. We observe that the resource optimal results are achieved when

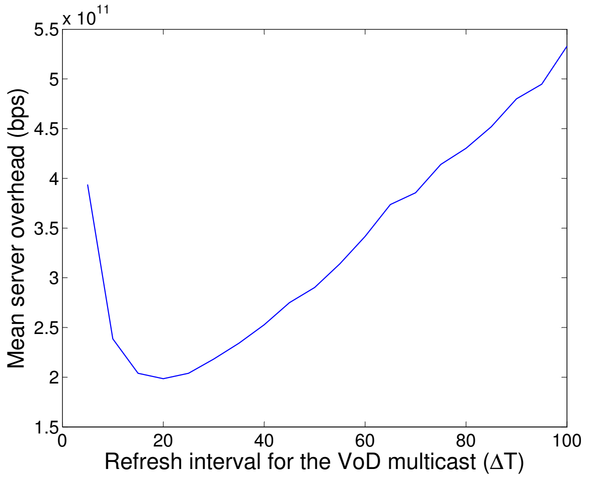


Fig. 3. Bandwidth utilization at the dedicated server when $L_S = 100$ minutes.

ΔT is around 20 $minutes$ (*i.e.*, $\Delta T^* \approx 20$). Based on these results, we can explain the relationship between servicing overhead and ΔT parameter as follows:

- $\Delta T < \Delta T^* \rightarrow$ the number of multicast streams for the given session is increased unnecessarily, *causing multicast portion of the server traffic to dominate the overall servicing overhead at the server side.*
- $\Delta T > \Delta T^* \rightarrow$ the desired usage rate for the session peers during synchronization phase is increased unnecessarily, *forcing the dedicated server to increase its unicast-based servicing rate faster than it decreases the multicast-based servicing rate.*

Furthermore, when the interarrival times between consecutive join requests are exponentially distributed, we can approximate the mean servicing overhead at the dedicated server as follows:

$$\begin{aligned} E[W_S] &= \frac{W_M}{4\rho^2} \times \Bigg[\frac{(\rho-2)(\rho-1)}{\mu_S/\Delta T} + \big(2\rho^2+\rho-4\big) \\ &\quad + \frac{\rho^2(L_S - 2\mu_S) + \mu_S(3\rho-2)}{\Delta T/2}\Bigg] \end{aligned} \tag{12}$$

where μ_S equals $1/\lambda_S$ and ρ equals $W_M/E[W_u]$. [14]

We show the theoretical results in Figure 4(a) as we vary the value assigned to ρ. The results coincide nicely with our simulation-based results that assume $\rho = 3$.

Then, using (12) we can determine the resource optimal ΔT parameter as follows:

$$\Delta T^* = \min\left(\sqrt{\frac{2L_S \times \rho^2/\lambda_S}{(\rho-2)(\rho-1)}}, L_S\right) \tag{13}$$

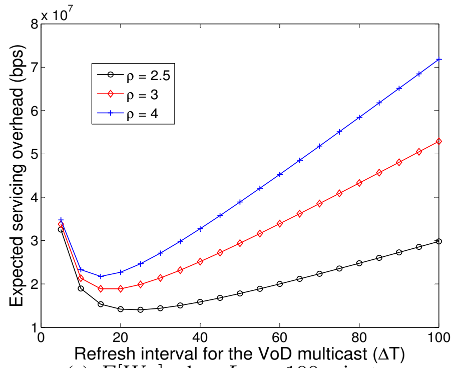
In Figures 4(b) and 4(c), we illustrate the relationship between the resource optimal ΔT^* parameter and the session length. We observe that, for all the considered scenarios, as we increase the value of L_S, the ratio of $\Delta T^*/L_S$ starts to converge, where the point of convergence is inversely proportional to the value of ρ. Furthermore, the value of ΔT that achieves the minimum servicing overhead introduces a fairly limited increase in overhead at the client side. Therefore, the selected ΔT^* value presents a good tradeoff point between server overhead and peer contribution.

[11] W_u represents the available bandwidth at the user side to service the received requests. In reality, to prevent local congestion, the value of W_u is expected to be less than the uplink servicing capacity at the user side. However, for the sake of simplicity, we assume their values to be equal.

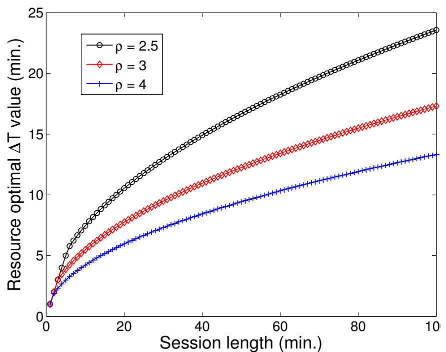
[12] Since a request that arrives at t needs to be serviced by $2 \times (t - T_1)$, where T_1 represents the starting point for the session's multicast, the maximum delivery duration for a session is given by $2 \times L_S \times W_M$, when ΔT attains the highest possible value of L_S.

[13] Worst-case scenario is attained when only one peer delivers data, whereas, best-case scenario is attained when all peers deliver the same amount of data.

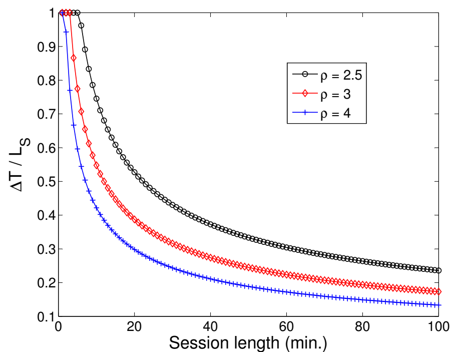
[14] Due to space constraints, we only present the results that correspond to the following scenario: $\rho > 2$.



(a) $E[W_S]$ when $L_S = 100$ minutes

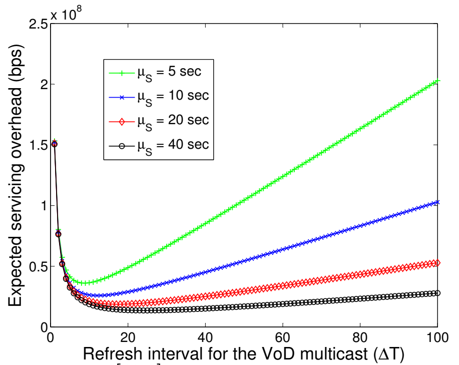


(b) ΔT^* vs. L_S

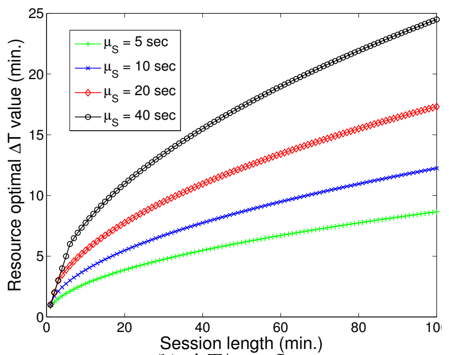


(c) Ratio between ΔT^* and L_S

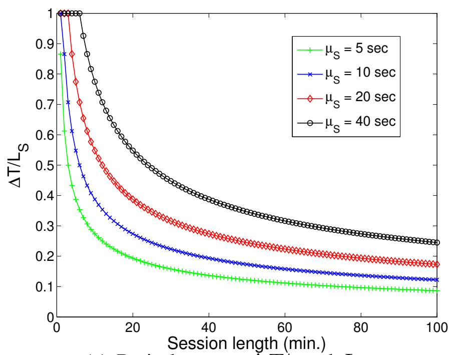
Fig. 4. Theoretical approximations on servicing overhead and refresh interval at different ρ values, when $\mu_S = 20s$.



(a) $E[W_S]$ when $L_S = 100$ minutes



(b) ΔT^* vs. L_S



(c) Ratio between ΔT^* and L_S

Fig. 5. Theoretical approximations on servicing overhead and refresh interval at different μ_S values, when $\rho = 3$.

Another advantage of the proposed framework is that it scales reasonably well in the number of requests. We show the results that correspond to the impact of varying request arrival rate in Figure 5(a). We observe that the minimum servicing overhead at the server side increases at a rate proportional to $\approx k^{\alpha}$, where k represents the rate of increase observed in the request arrival rate and α represents the scaling factor with a value of ≈ 0.46. For instance, increasing λ_S by 100 leads to an ≈ 8.5 times increase in the servicing overhead. Furthermore, as shown in Figures 5(b) and 5(c), increasing the request arrival rate also causes a noticeable decrease in the ΔT^* and $\Delta T^*/L_S$ values, *e.g.*, doubling λ_S reduces ΔT^* by 30%. As a result, we also observe a noticeable decrease in the peer capacity usage. That is because, even though the peer capacity available per request does not change, the amount of data that needs to be delivered per request reduces at the same ratio (around 30%).

Another important statistical measure for the proposed framework is the distribution for the maximum bandwidth usage at the server side (per session), for which the results are shown in Figure 6(a) for the unicast capacity usage and in Figure 6(b) for the overall capacity usage (including both unicast and multicast). Since the maximum bandwidth usage at the server side illustrates the worst-case resource allocation scenario (as perceived by the service provider), we need to carefully examine the distribution of such values so as to optimize the servicing cost at the server side, especially when multiple sessions are taken into consideration. The simulation results suggest that, in general, capacity usage shows an increasing trend as the value of ΔT increases. However, the results do not deviate significantly from the mean, suggesting a stable performance.

We next study the relationship between maximum bandwidth usage at the server side and the mean servicing overhead, for which the results are shown in Figure 7. We observe that as we increase the value of ΔT, max-to-mean usage ratio initially experiences a sharp increase in value, after which it starts to converge, to a value of 3.5. The results also illustrate the relative impact of unicast- and multicast-based overhead on the overall server capacity usage. Note that, since the max-to-mean server capacity usage ratio experiences a converging trend, depending on the selected ΔT value, service provider can effectively use this relationship to pre-allocate its resources to different sessions with the objective of optimizing the overall capacity usage in the network and minimizing the overall servicing cost.

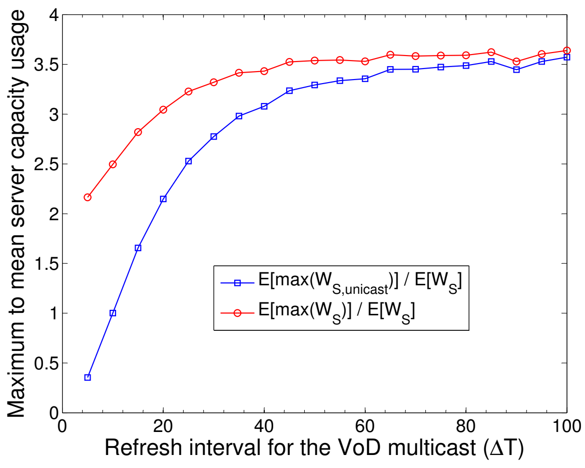
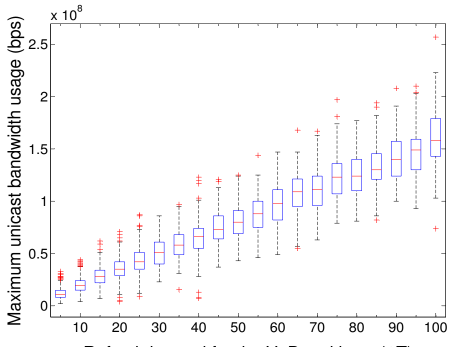
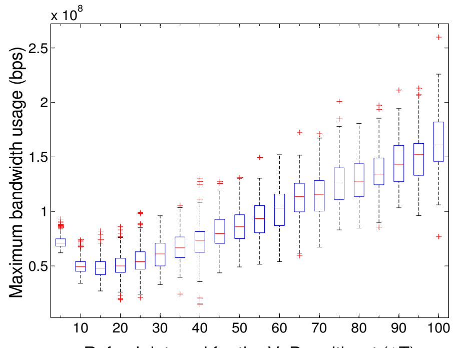


Fig. 7. Max-to-mean server capacity usage ratio.

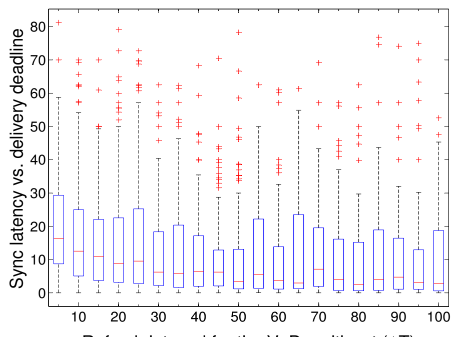
We next investigate the synchronization latency performance of the C-WFQ approach. In our simulations, we observe that less



(a) Unicast servicing overhead



(b) Overall servicing overhead



(c) Earliness measure for the processed requests

Fig. 6. Distributions for the servicing overhead and the latency measure.

than 1% of requests are delivered earlier to the clients. [15] In other words, almost all the requests are delivered at the request deadline, which coincides with the objectives stated for the C-WFQ approach. Specifically, since the C-WFQ approach aims to deliver the requested packets within the deadline by distributing peer resources to multiple requests at once, oftentimes minimal resources are utilized (at the peers) for each active request to allow for efficient servicing of the received requests. Additionally, for the requests that are processed earlier, Figure 6(c) shows the variations observed in the perceived latency values as we vary ΔT. We observe that as we increase the value of ΔT, we reduce the synchronization latency (from $\approx 15\%$ at $\Delta T = 5\ min.$ to $\approx 3\%$ at $\Delta T = L_S$), which is expected as increasing the value of ΔT also increases the total amount of data that needs to be delivered per session peer. As the session peers hit the the uplink capacity barrier faster at higher ΔT values, session peers start to reduce the bandwidth they allocate per request to a value around or less than the minimum required servicing bandwidth for the given request. As a result, with the help of the dedicated server, most of the received requests are serviced at the minimum required delivery rate.

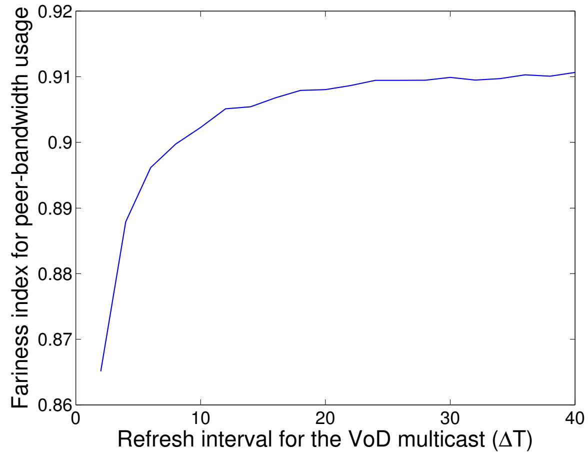


Fig. 8. User fairness results when $L_{VoD} = 100$ minutes.

Lastly, we illustrate the fairness performance of the C-WFQ approach in Figure 8. In general, we observe fairly good results for the fairness performance, which attains a value of above 0.9 when $\Delta T \approx \Delta T^*$. The results improve as we increase the value of ΔT, however the level of improvement is limited above ΔT^* as the values start to converge (*e.g.*, converge to a value of ≈ 0.91 in Figure 8).

[15] Specifically, we find that on average $9.26 \times 10^{-2}\%$ of the received requests are delivered earlier, when the C-WFQ approach is used.

V. Conclusion

In this paper, we proposed a novel cooperative delivery framework for the delivery of on demand content in IPTV networks. To achieve the desired performance improvements in network scalability, we utilized session peers for content distribution. To improve the fairness performance for the peer contributions, we proposed a dynamic resource allocation scheme that adapts the delivery rates for the received requests depending on the earlier contributions and the user weights. Through extensive simulation studies, we analyzed the most critical performance measures for the proposed framework and showed significant improvements in servicing overhead, while achieving very good fairness results.

References

[1] W.-Pk. Yiu, Xing Jin, and S.-H.G. Chan, "Challenges and approaches in large-scale p2p media streaming," *Multimedia, IEEE*, vol. 14, no. 2, pp. 50 –59, april-june 2007.

[2] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and K.W. Ross, "A measurement study of a large-scale p2p iptv system," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1672 –1687, dec. 2007.

[3] Yang Xiao, Xiaojiang Du, Jingyuan Zhang, Fei Hu, and S. Guizani, "Internet protocol television (IPTV): The killer application for the next-generation Internet," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 126–134, November 2007.

[4] I. Bermudez, M. Mellia, and M. Meo, "Investigating overlay topologies and dynamics of p2p-tv systems: The case of sopcast," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1863 –1871, october 2011.

[5] K. Kerpez, D. Waring, G. Lapiotis, J.B. Lyles, and R. Vaidyanathan, "IPTV service assurance," *IEEE Communications Magazine*, vol. 44, no. 9, pp. 166–172, 2006.

[6] P. Diminico, V. Gopalakrishnan, R. Jana, K.K. Ramakrishnan, D.F. Swayne, and V.A. Vaishampayan, "Capacity requirements for on-demand iptv services," in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, jan. 2011, pp. 1 –10.

[7] Cristiano Costa, Italo Cunha, Alex Borges, Claudiney Ramos, Marcus Rocha, Jussara Almeida, and Berthier Ribeiro-Neto, "Analyzing client interactivity in streaming media," 2004.

[8] M. Vilas, X.G. Paneda, R. Garcia, D. Melendi, and V.G. Garcia, "User behavior analysis of a video-on-demand service with a wide variety of subjects and lengths," in *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*, aug.-3 sept. 2005, pp. 330 – 337.

[9] Weimin Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *In Proc. of ACM EuroSys*, 2006, pp. 333–344.

[10] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu, "Modeling channel popularity dynamics in a large IPTV system," in *ACM SIGGMETRICS/Performance'09*, 2009.

[11] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperative diversity Part I and Part II," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1927–1948, Nov 2003.

[12] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC Research Report TR-301, Sep 1984.