Maximizing Streaming Flows Based on a Novel Video Streaming Framework

Tian Wang
College of Computer Science
and Technology
Huaqiao University
Xiamen, Fujian Province
Email: wsnman@gmail.com

Weijia Jia
Department of Computer Science
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong
Email: itjia@cityu.edu.hk

Bineng Zhong
College of Computer Science
and Technology
Huaqiao University
Xiamen, Fujian Province
Email: bnzhong@gmail.com

***Abstract*—Video streaming is a kind of bandwidth hungry application. As a consequence, the number of streaming flows may be restricted. In this paper, a novel video streaming framework is designed, where multiple NVSs (Network Video Servers) form into a server group to collaboratively provide quality services. A novel problem – Maximum Streaming Flows (MSF), aiming to maximize the number of simultaneously online users is proposed. This problem is proved to be NP-Complete and can be simplified to MSF-2 by adding relays restriction. We design a $(1-\epsilon)$ approximation algorithm, where ϵ is a constant which tends to be infinitesimal with the increasing number of successful streamed flows. We conduct extensive simulations to show the effectiveness of the methods proposed as compared with several traditional solutions.**

I. Introduction

Video streaming consumes lots of bandwidth compared with other data such as audio or web data. There are mainly two general architectures for real-time video streaming system. The first one is client-server (C/S) scheme where the data sources stream the data captured to a server first and remote users then retrieve the video from the server. This scheme is simple but incurs a heavy burden on the servers and does not scale well. By contrast, Peer-to-Peer (P2P) streaming scheme greatly alleviates the burdens of servers with good scalability. However, one of the problems about P2P scheme is that the video sources act as both clients and servers which incurs burden for sources such as IP cameras. Moreover, it is known that media streaming over best-effort packet networks such as the Internet or wireless networks is quite challenging because of some of the dynamic and unpredictable factors such as available bandwidth, loss rate, and delay [1]. If the current connection is unstable, the media flows cannot be streamed to users or the QoS is not acceptable.

To address these problems, we proposed a novel video streaming framework. The basic idea is that several video servers cooperatively form a server group to provide both streaming and storage services. Different from the traditional C/S architecture, multiple servers cooperatively undertake the streaming task. The proposed new streaming scheme can fully utilize the "server diversity" of dynamic networks. It has been shown in [2] that usage of multiple streaming servers provides better robustness in case one of the channels becomes congested. We expect that multiple servers and path diversity help in achieving higher overall throughput to the end users.

We mainly make the following contributions: We propose a novel video streaming framework which introduces multiple video servers to collaboratively provide services for end users. Based on this framework, we formulate a novel problem: Maximum Streaming Flows (MSF) , which aims to maximize the total number of simultaneous online users. This problem is proved to be NP-complete. For applications in reality, we add two-hops relay restriction and get the simplified MSF-2 problem. We design a approximation algorithm with a provable performance bound compared with the optimal solution.

II. Related Work

Video streaming aims at providing high quality video content to users of both live and on-demand services. Traditional client-server based video streaming solutions incur expensive bandwidth provision cost on the server and are not scale well [3]. On the other hand, Peer-to-Peer (P2P) streaming greatly alleviates the burdens of servers with good scalability [4]. In these typical P2P streaming systems, users named as peers act as both clients and servers, which incurs burden for sources such as IP cameras. Recently, cloud computing is redefining the way many Internet services are operated and provided, including video streaming. Paper [5] proposes a predictive cloud system that dynamically books the minimum bandwidth resources from multiple data centers for the VoD provider. Our proposed framework is also a cloud computing streaming architecture and can dynamically exploit bandwidth resources.

The streaming framework proposed in this paper also takes advantage of the "server diversity" [2]because original video sources can pick out better servers to act as steaming servers. Our work is different from [2] where one source splits streaming media into several sub-streams and each sub-stream is delivered separately to the individual users. In our work we assume one flow is streamed through an exclusive path which reduces the complexity of implementation. It makes our problem be different from other problems such as Multi-commodity problem and traditional routing algorithm [6] cannot be used.

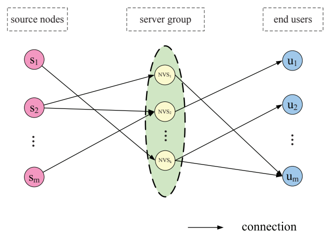

Fig. 1. Video streaming framework.

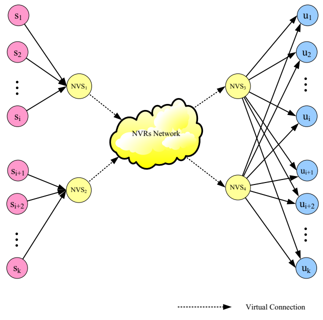

Fig. 2. A special case of MSF problem.

III. Multiple Streaming Servers Scheme

The streaming architecture is shown in Fig. 1 which includes tripartite sides: sources, servers and end users. Sources are the origins of video streams such as IP cameras while end users are requesting these video flows. Multiple servers connect each other in the networks. In this paper, we also refer servers as NVSs (Network Video Servers), since NVSs with large-capacity hard disks can implement storage of mass video data [7]. Video flows are first streamed to one of the NVSs for the purpose of storage as well as for later retrievals and then will be further streamed directly to end users or via other NVSs for further relay. Multiple NVSs can cooperatively forward this streaming flow based on the exchanges of available bandwidth information between themselves. The advantages for relays are two-fold. The one is to exploit "server diversity" to forward video flows while the other is to produce flow copies at multiple servers for backups.

Our objective is to maximize the total number of video flows streamed to users. For each flow requested by the user, we assume that the available bandwidth of the transmission link should greater than a fixed data rate f_i. In this paper, we use f_i to denote a flow itself as well as its bandwidth requirement. Another assumption is that NVSs can measure the available bandwidth of the link from sources and to users as well as the available bandwidth between themselves [8].

Before formulating the problem, we first introduce a definition below.

Definition 1: Flow Integrality (FI) constraint. When we say a streaming flow is under FI constraint, it means that this flow must be sent from one source to the destination via only one path.

We now formally define the general Maximum Streaming Flows (MSF) Problem problem.

Definition 2: MSF Problem - Let a set of source nodes be $S = \{s_1, s_2, ..., s_m\}$, a group of servers be $NVSs = \{NVS_1, NVS_2, ..., NVS_h\}$ and a set of corresponding destination users be $U = \{u_1, u_2, ..., u_m\}$. These source and user nodes are connected to NVSs through wired or wireless networks and NVSs are also connected to each other. Source nodes stream flows to NVSs and users retrieve flows from NVSs. Each link (u, v) connected to the NVS is associated with a capacity constraint $c_(u, v)$. Each s_i will produce a video flow under the bandwidth requirement f_i. One user u_i is expecting/requesting the media service from a specific source s_i. The flow delivered to the user should be under the FI constraint and flows can be relayed among NVSs arbitrarily. The objective is to maximize the total number of users (flows) simultaneously served.

The decision version of the problem is to ask whether a subset of these flows can go through the network to corresponding users while the number of users simultaneously served is greater than a constant C. In the next, we prove its NP-completeness.

Theorem 1: **The decision version of the MSF problem is NP-Complete.**

Proof: Obviously, this problem is in NP. In the next, we only prove its NP-hardness.

To prove the problem is NP-hard, we consider a special case of the original problem which is shown in Fig.2. There are k source nodes, either connected to NVS_1 or NVS_2. Suppose the flow $f_k = 1$ for any k and all link capacities between sources and NVSs are 1. On the other side, there are k users, which are all connected to NVS_3 and NVS_4. The link capacities between users and NVSs are also 1. These four NVSs are connected by NVSs networks, which consists of multiple NVSs. The "virtual connection" means there may be multiple connection paths between the two sides. Obviously, for this special case, the problem can be described as: whether we can forward these k flows from NVS_1 and NVS_2 to the destination NVS_3 and NVS_4, under the flow capacity constraints. This problem is exactly the D2CIF problem, which is proved to be NP-hard [9]. ■

IV. Algorithms for real applications

Since the MSF problem is NP-complete, the polynomial time algorithm cannot be found. Therefore, we have to seek to simplify the problem for practical solution. We note that in reality, if there are too many relay hops among the NVSs it will lead to both long delay and high loss rate which may not satisfy the real-time requirement. Moreover, multi-hop relays among NVSs will lead to extra communication cost between NVSs. We then simplify MSF problem to MSF-2 problem by permitting relays by only two servers(NVRs). To be consistent with MSF-2, we denote the streaming scheme as MSF-0 if there are no intermediate relays between the source and the corresponding user. Moreover, the general MSF problem is denoted by MSF-n since multiple relays may be considered.

We denote the flows by $f_1, f_2, ..., f_m$. Without loss of generality, we assume these flows are sorted by nondecreasing order. For each link l_k which is between two NVSs (denoted by NVS_i and NVS_j), denote the set of flows that may be served by link l_k by $\lambda(l_k)$. Flow f_q belongs to set of $\lambda(l_k)$ if the following three requirements are satisfied: 1) There are adequate bandwidth for source node s_q to NVS_i; 2) The bandwidth of link l_k is greater than f_q; 3) There are adequate bandwidth from NVS_j to end user u_q. For each link l_k, we define a variable $f^*_{l_k}$. Initially, all the $f^*_{l_k} = NULL$. During

Algorithm 1 MSF-2 algorithm

Input: a flow set $\{f_i\}$; links among NVSs: $\{l_k\}$; $B(l_k)$ which refers to the available bandwidth left for l_k and the corresponding $\{\lambda(l_k)\}$ which refers to the flows may be streamed by that link (candidate flows).
Output: N_u-The number of users can be served; $\Omega(l_k)$-The flows assigned to link l_k

```
1: $N_u = 0; i = 0;$
2: sort the flows in the order of nondecreasing by their sizes: $f_1, f_2, ..., f_m$.
3: for $i = 1 \rightarrow m$ do
4:   if $f_i$ can be served by one of the links $(l_s)$ then
5:     assign $f_i$ to link $l_s$ (i.e. add $f_i$ to $\Omega(l_s)$) and $B(l_s) = B(l_s) - f_i$;
6:   else if exist some link $l_k$ who still has available bandwidth (i.e. $f^*_{l_k} == NULL$) && $f_i \in \lambda(l_k)$ then
7:     assign $f_i$ to link $l_k$ (i.e. add $f_i$ to $\Omega(l_k)$) and $B(l_k) = B(l_k) - f_i$;
8:     $f^*_{l_k} = f_i$;
9:   else
10:    for the link $l_k$ who have no available bandwidth left (i.e. $f^*_{l_k} \neq NULL$) do
11:      assign any of the flow $f_\theta$ already assigned to any other link $l_p$ who still has available bandwidth (i.e. $f^*_{l_p} == NULL$) && $f_\theta \in \lambda(l_p)$;
12:      $B(l_k) = B(l_k) + f_\theta$; $B(l_p) = B(l_p) - f_\theta$;
13:      if the link $l_p$ has no available bandwidth then
14:        The maximum flow assigned to $l_p$ becomes its $f^*_{l_p}$;
15:      end if
16:      if the link $l_k$ still has available bandwidth then
17:        assign $f_i$ to $l_k$ (i.e. add $f_i$ to $\Omega(l_k)$) and $B(l_k) = B(l_k) - f_i$;
18:        $f^*_{l_k} = f_i$;
19:        GOTO step 3;
20:      end if
21:    end for
22:    $N_{NS} = N_{NS} + 1$; //$f_i$ can not be assigned
23:  end if
24: end for
25: $N_u = m -$ | flow whose $f^* \neq NULL| - N_{NS}$;
26: Output $N_u$ and $\Omega(l_k)$ for each link $l_k$;
```

the running of the algorithm, there are two status for one link l_k: not-full or full, where full means that there is no available bandwidth left (i.e. its $f^* \neq NULL$), otherwise is not full ($f^* == NULL$). If we assign f_i to one link and then this link becomes full, we set $f^*_{l_k} = f_i$. We assign flows to the link between NVSs one by one. Our basic idea is that, one flow f_i cannot be assigned if and only if all the link l_k where $f_i \in \lambda(l_k)$ is full and the flows already assigned to l_k do not belong to any $\lambda(l_p)$ where the state of link l_p is not-full. The algorithm is shown in Algorithm 1 in detail.

Lemma 1: **After running the algorithm, the size of the flow not assigned (denoted as f_k) is no smaller than any of flow assigned to the link l_k where $f_k \in \lambda(l_k)$.**

Proof: For the link l_k which $f_k \in \lambda(l_k)$, there must be a $f^*_{l_k} = f_b$ according to our algorithm otherwise f_k will become the $f^*_{l_k}$. Obviously, f_b is the maximum flow in the link l_k. If f_k is smaller than f_b, according to our algorithm f_k will be prior to f_b to become the $f^*_{l_k}$, which contradicts to our assumptions. ■

Lemma 2: **After running the algorithm, if all the links are full, denote N_{opt} as the optimal solution of our problem, then $N_u \geq N_{opt} - n_l$, where n_l is the number of links among NVSs.**

Proof: Obviously, after the running of our algorithm, it is impossible to assign any flow f_k, which was not assigned, to any links without removing the flow already assigned. That is, if we add any of the flows not assigned to any link, then some flow already assigned must be removed. Moreover, according to Lemma 1, f_k is larger than any of the flows which were assigned to the link l_p where $f_k \in \lambda(l_p)$. If we assign f_k to any l_p, it means that some $f_p \leq f_k$ must be removed from that link, which will not increase the total number of flows assigned. For each link, there is at most one flow which is assigned but cannot be served and our output N_u is the number of flow that can be served. Obviously, the optimal method can serve at most all the flows already assigned, which means $N_u \geq N_{opt} - n_l$. ■

Theorem 2: **Algorithm 1 is a $1 - \epsilon$ approximation algorithm, where ϵ is equal to $\frac{n_l}{N_{opt}}$. The computation time is $O(m^2 + 2mn^2)$, where m is the total number of flows and n is the number of NVSs (servers).**

Proof: The flows can be divided into two subset F_x and F_y, where F_x is the subset containing the flows assigned to the links which are not-full (these links are denoted as L_x) and F_y contains the flows assigned to the links which are full and the flows not assigned (these links are denoted as L_y). Thus the original problem can be divided into two subproblems x and y, whose input flows are F_x and F_y respectively. According to our algorithm, all of the flows in F_x can be served and the flows in F_y are not in any $\lambda(l_k)$ where l_k is the link in L_x. Denote our output for solving the subproblem x by N_{u_1}. Obviously, the optimal solution can be denoted as

$$N_{opt} = N_{opt_x} + N_{opt_y} \tag{1}$$

where N_{opt_x} is the optimal solution for the subproblem x while N_{opt_y} is the optimal solution for the subproblem y. Obviously,

$$N_{opt_x} = |F_x| \tag{2}$$

where $|F_x|$ is the cardinality of F_x. According to Lemma 2, we have:

$$N_{u_2} \geq N_{opt_y} - |L_y| \tag{3}$$

where $|L_y|$ is the cardinality of link set L_y. Then, we have:

$$\begin{aligned} N_u &= |F_x| + N_{u_2} \\ &\geq N_{opt_x} + N_{opt_y} - |L_y| \end{aligned} \tag{4}$$

According to Equ. (1) and $|L_y| \leq n_l$, we have:

$$\begin{aligned} \frac{N_u}{N_{opt}} &\geq \frac{N_{opt_x} + N_{opt_y} - |L_y|}{N_{opt}} \\ &\geq \frac{N_{opt} - n_l}{N_{opt}} \\ &= 1 - \epsilon \end{aligned} \tag{5}$$

, where ϵ is equal to $\frac{n_l}{N_{opt}}$ and can be infinitesimally small with the increasing of simultaneous online users. We also draw a figure to show the value of ϵ in Section V.

Obviously, the sorting cost of flows is $O(m^2)$. For a flow f_i, it will take $O(n_l)$ time to find which link can fully serve the flow. Since there are m flows, it will take mn_l time for the executions of step 5 to step 7. For the executions of step 9 to step 23, the key operation is to shift flows from one link to another. We note that in the worst case the m flows all belong to $\lambda(l_k)$ where l_k is one of the n_l links. Hence, for one link, the maximum number of flows assigned is no greater than m, which results in the total number of assignments/shifting for flows is no greater than mn_l. Therefore, the total computation time of the algorithm is $O(m^2) + 2mn_l = O(m^2 + 2mn_l)$. Moreover, the total number of links n_l can be represented by $(n-1)^2$ where n is the number total NVSs. In conclusion, $O(m^2 + 2mn_l) = O(m^2 + 2mn^2)$. ■

V. Performance Evaluations

This section presents the evaluation of the algorithms using MATLAB. We compare the proposed methods(MSF-2) with the traditional "one-server" method and P2P method (MSF-0). As a baseline, we also run the brute-force method to get the optimal solution when multi-hop relays are allowed among servers, which is denoted as MSF-n. In the simulations, by saying "P2P bandwidth", we mean the bandwidths from data sources to servers or the bandwidths from servers to users. To simulate the dynamics of network bandwidths, we assume the P2P bandwidth obeys log-normal distribution [10] and so do the video flows since based on Kun-chan's [11] research flow rates can often be described by a log-normal distribution.

Fig. 3 shows the numbers of failed media flows when the mean value of P2P bandwidth varies from 800 to 1000. In this simulation, we set the number of servers (NVSs) by 3. This figure shows that all methods yield less numbers of failed flows when the P2P bandwidth is increased. This is because, with the increase of bandwidth more flows can be streamed from sources to users. The MSF-2 methods proposed always yields the best performance among all algorithms, which validates the effectiveness of the proposed server group streaming structure as well as the algorithm proposed. We also simulate the scenarios if relays between servers are not permitted. This figure shows that there is little difference between the MSF-2 and MSF-n, which means that one-hop relay among servers is enough for most of flows.

Fig. 4 shows the number of failed flows when the deviation of P2P bandwidth increases from 10000 to 50000, which means the bandwidth fluctuations becomes more and more intensely. As expected, the numbers of failed flows are growing for all solutions, which means the dynamics of networks would not be a good thing for streaming applications. The good thing is that our proposed methods are more tolerant of this network dynamics, at least much better than traditional C/S and P2P methods. Especially, even when the deviation of P2P bandwidth reaches 50000, the number of failed flow is just 9. These simulation results show that our method is better to be applied in dynamic networks, such as wireless or mobile environments.

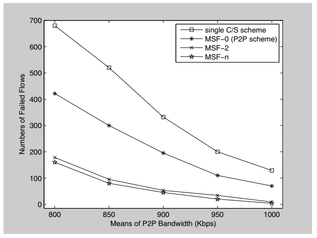

Fig. 3. The number of failed flows vs. P2P bandwidths

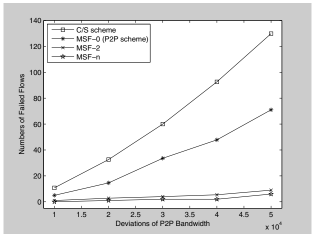

Fig. 4. The number of failed flows vs. the deviations of P2P bandwidth

VI. Conclusions

In this paper, we studied the streaming of realtime video flows problem. We designed a novel video streaming framework where Multiple NVSs form a server group to collaboratively provide services to remote users. Our streaming framework combined the advantage of both traditional client/server scheme and P2P scheme. For one thing a server group acts as the steaming servers which are more powerful. For another, multiple servers grouped can shift the streaming loads among them. The simulation results showed that our video streaming scheme plus the designed approximation algorithm can provide better services and support more simultaneous online users.

Acknowledgment

Above work is supported in part by grants from the General Research Fund of the Hong Kong SAR, China No. (CityU 114012, CityU 114513); and CityU Applied R & D Grants (ARD) No. 9681001; (ARG) No. 9667052; NSF (China) No. 61070222, No. 61202468, No. 61202299 and Shenzhen (China) Basic Research Project No. JCYJ20120618115257259; the Starting Research Fund from Huaqiao University under Grant No. 12BS218 and the Fundamental Research Funds for the Central Universities under Grant No. JB-ZR1220.

References

[1] J. G. Apostolopoulos and M. D. Trott, "Path diversity for enhanced media streaming," *IEEE Communications Magazine*, vol. 42, pp. 80–87, 2004.

[2] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proceedings of IEEE Infocom*, 2002, pp. 1736–1745.

[3] X. Zhu, H. Deng, Z. Chen, and H. Yang, "Design of large-scale video surveillance system based on p2p streaming," in *3rd International Workshop on Intelligent Systems and Applications (ISA)*, 2011, pp. 1–4.

[4] C. Wu, B. Li, and S. Zhao, "Diagnosing network-wide p2p live streaming inefficiencies," in *infocom*, 2009, pp. 2731–2735.

[5] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *INFOCOM'12*, 2012, pp. 460–468.

[6] M. Pioro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Elsevier Inc, 2004.

[7] A. Girgensohn, D. Kimber, J. Vaughan, T. Yang, F. Shipman, and T. T. E. Rieffel, "Dots: Support for effective video surveillance," in *Proceedings of the 15th international conference on Multimedia*, 2007, pp. 423–432.

[8] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *In Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002, pp. 14–25.

[9] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing (SIAM)*, vol. 5, no. 4, pp. 691–703, 1976.

[10] H. Balakrishnan, S. Seshan, M. Stemm, and R. Katz, "Analyzing Stability in Wide-Area Network Performance," in *Proceedings of ACM SIGMETRICS*, Seattle, WA, Jun. 1997, pp. 2–12.

[11] K. chan Lan and J. Heidemann, "A measurement study of correlation of Internet flow characteristics," *Computer Networks*, vol. 50, no. 1, pp. 46–62, January 2006.