

Scalable and Reliable IPTV Service Through Collaborative Request Dispatching

Shicong Meng[†] Ling Liu[†] Jianwei Yin[‡]

[†]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
{smeng, lingliu}@cc.gatech.edu

[‡]College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, P.R.China
zjuyjw@cs.zju.edu.cn

Abstract

IPTV has emerged as the future standard of television and drawn enormous attention from both industry and research communities. Among different IPTV services, on-demand services are the most popular ones due to their convenience and rich content. However, supporting scalable and reliable on-demand IPTV services remains to be an important challenge. Existing IPTV architecture dedicates a centralized regional station to serve subscribers in the respective region regardless of temporal and spatial dynamics in service demand. As a result, it may cause significant imbalance of resource utilization and service provisioning delay at different stations, especially with increasing subscribers and video content.

In this paper, we propose to allow IPTV stations of different regions to collaboratively serve user requests for delivering scalable and reliable IPTV services. One key challenge in achieving this station-wise collaboration is to route service requests to appropriate stations according to cost-effectiveness and load distribution in a fully distributed manner. We devise a novel request dispatching protocol which runs on each IPTV station, and yet forms a collaborative dispatching strategy that avoids hot spots and reduces service delivery cost at the same time. Our experiment results suggest that our service request dispatching algorithm significantly improves the scalability of on-demand IPTV services for the existing IPTV architecture.

1 Introduction

With the rapid advance of Internet technology and the steady grows of Internet end users' bandwidth, Internet Protocol Television(IPTV) becomes increasingly popular worldwide as a new convenient way of providing commercial-grade live broadcasting TV, Video on-Demand(VoD), on-Demand Programming(oDP) and other continuous content streaming services. The number of IPTV subscribers has reached 4.3 million in 2005 and is estimated to increase to nearly 60 million by 2011 [14]. One of the most attractive technology advantages of Internet Protocol TV is unicast streaming, which enables it to offer on-demand everything and contributes to an increasingly large portion of the total traffic [12].

To deliver on-demand IPTV services, service providers

push on-demand content from their nationwide datacenters to a large number of local service stations at different geographical regions. These regional stations perform the last mile service delivery by exclusively serving all requests of their local IPTV service subscribers. This architecture, however, faces a number of challenges given ever increasing subscribers and service catalogs. First, as each station exclusively serves subscribers in its region, i.e. an one-to-many mapping, each station essentially becomes a single-point-of-failure. Second, due to high dynamics of service demand in both location and time [7, 15, 19], each station has to be heavily over-provisioned to ensure consistent service quality. Last but not the least, we argue that the one-to-many mapping architecture fails to explore cost sharing opportunities across regional stations to reduce per-view resource consumption and ultimately scale the delivery of on-demand IPTV services.

One promising approach to address this problem is to organize IPTV service stations into collaborative groups, where a subscriber can be served by multiple station nodes and a station can serve subscribers in multiple regions. This many-to-many subscriber-to-station mapping not only avoids single-point-of-failure but also offers opportunities for inter-node cost-sharing. Nevertheless, one fundamental problem in this station-wise collaboration is service request dispatching, i.e. determining which station in a collaborative group to serve an incoming request. The dispatching process should consider the cost-effectiveness of request serving and load balance. More importantly, the dispatching mechanism should be scalable and reliable given the massive request rate of IPTV service and the potential business loss caused by system downtime.

In this paper, we present such an IPTV service delivery framework, iCloud, which organizes regional stations into *service groups* by utilizing IPTV network connectivity characteristics. To address the service request dispatching problem, iCloud runs a utility-based dispatching protocol that allows nodes within each service group to independently make request dispatching decisions, aiming at minimizing per-view resource consumption and reducing the impact of hotspots on request serving. Our request dispatching is fully distributed, which enables service groups to scale out easily by simply adding more service stations, and minimizes the impact of station failure. Furthermore, our theoretical

analysis shows that the dispatching algorithm can quickly reach stable dispatching, even though it does not require global coordination. Our experimental results also show that iCloud delivers significantly higher service throughput compared with the existing architecture.

The remainder of this paper is organized as follows: In Section 2, we first introduce current IPTV service architecture and outline the problem statement, then we present an overview of iCloud overlay. Section 3 describes the request dispatching protocol in iCloud. Section 4 evaluates and characterizes iCloud in terms of performance, scalability as well as load balance. We summarize the related work in Section 5 and conclude the paper in Section 6.

2 Overview

In this section we first review the basics of current IPTV services. We then outline several problems with the current IPTV service architecture, and highlight the design of iCloud.

2.1 Current IPTV Service Architecture

Similar to video clips that are broadcasted at YouTube, IPTV services deliver digital television, video on demand, personal video recorder as well as other continuous content streaming services to customers by using Internet Protocol over a network infrastructure, e.g. broadband connections provided by telephone companies.

As Figure 1(a) illustrates, current IPTV service architecture is hierarchical and consists of three parts, one or several nationwide *Super Hub Offices(SHOs)*, many *Video Hub Offices(VHOs)* for different regions and many *Set-top Boxes(STBs)* for each subscriber.

SHOs are usually datacenters and serve as the dissemination point of video content, such as broadcasting TV programs, movies for video on demand. SHOs break up video streams into IP packets and transmit them to geographically distributed VHOs with the Internet Group Management Protocol(IGMP). VHOs as regional service stations directly deliver IPTV service to their designated neighborhoods. VHOs support multicast-based TV broadcasting service via IGMP. They also deliver unicast on-demand services based on locally stored video content. Due to the enormous amount of video content and limited resources available at each VHO, each VHO can only store a portion of the total video content locally and the set of video content may vary from VHO to VHO. When certain content requested is not locally available, a VHO needs to download it from a SHO. VHO nodes and SHO nodes are connected via high speed backbone network which provides sufficient bandwidth for inter-node video transmission, although point-to-point bandwidth between far away VHO nodes may be limited. Each IPTV subscriber is equipped with a set-top box which connects to the home Digital Subscriber Line(DSL), and is responsible for reassembling packets into a coherent video stream and then decoding the content.

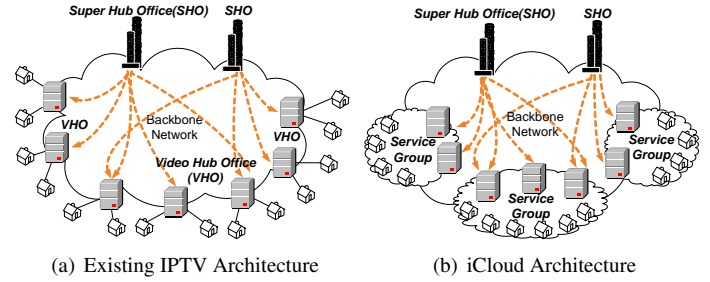


Figure 1. IPTV Service Architecture

On-demand IPTV services gain more and more popularity and contributes a significant portion of today's IPTV traffic [12]. As they rely on resource-intensive unicast streaming, the capacity of on-demand services largely determines the scale of IPTV systems. Therefore, we focus our discussion on efficient on-demand IPTV service provisioning through inter-station collaboration.

2.2 Problem Statement

Most IPTV service providers adopt the one-to-many service architecture such that one regional station is responsible for delivering on-demand IPTV services to many subscribers in the respective region and each subscriber can be served only from one pre-specified regional station. We argue that this architecture has a number of critical drawbacks.

First of all, serving subscribers of a given neighborhood region with a single service station makes the station a single-point-of-failure, especially when a sudden surge of service demand or unexpected failure at the station causes servers to be unavailable. Second, previous measurement studies [7, 15, 19] reveal that on-demand services often experience heavily skewed load both in temporal and spatial dimensions. For example, certain stations may undertake overwhelming workload while other stations observe relatively little load during the same time period. Furthermore, such skewness may dramatically change from time to time and move from location to location. In general this skewed load distribution may lead to overall under-utilization of resources and poor service quality at peak time. Last but not least, the isolated *request processing* limits opportunities for cost-sharing in the current architecture. For instance, comparing to the one-to-many mapping where each station serves requests from its local subscribers, a group of regional stations, when collaboratively serve their subscribers, can serve content more efficiently by aggregating many requests on the same content to fully utilize memory-based cache and reduce disk IO [16].

2.3 Overview of iCloud Design

One way to address the aforementioned problems is to introduce a many-to-many mapping between regional stations and subscribers such that a subscriber can access services provided by multiple stations and a station can serve sub-

scribers from multiple regions. This many-to-many mapping removes single-point-of-failure as a subscriber can access multiple stations. Furthermore, the many-to-many mapping also creates more opportunities for inter-station collaboration and cost-sharing.

Despite various advantages of many-to-many mapping, adapting existing IPTV service architecture to support inter-station collaboration is still challenging. It requires us to answer two fundamental questions. First, *how to determine the scope of collaboration?* Should we allow all nodes to collaborate with each other in a single service group, or organize them into multiple ones? These questions can be answered only after we understand the underlying factors in inter-station collaboration. Second, *how to dispatch IPTV service requests when there are multiple candidate VHO nodes? Can we reduce request serving cost through cost-sharing?* In general, given the scale of IPTV service and its ever increasing demand, a request dispatching approach should meet both scalability and cost-efficiency requirements.

To answer these questions, we present iCloud, an IPTV service framework which enables existing IPTV service architecture to support inter-station collaboration, and efficiently delivers on-demand IPTV services. Concretely, iCloud addresses the above challenges with the following two techniques:

Proximity-aware Service Group Formation. iCloud organizes regional stations into service groups to support intra-group collaboration among stations. IPTV service is bandwidth sensitive and the available bandwidth between two stations often drops when the number of hops between them increases. Hence, determining the size of service group should consider the corresponding impact on inter-station bandwidth. iCloud employs a proximity-aware service group formation protocol which ensures two properties: (1) A service group has enough station nodes for collaboration; (2) Any two stations within a service group are at most k hops away from each other to ensure adequate inter-station bandwidth between them, where k is a configurable parameter. The group formation protocol iteratively merges small groups into large ones and has fast convergence speed. It also allows advanced tuning such as non-uniform k as service providers often have deep knowledge of their service delivery network in different regions.

Utility-Driven Request Dispatching. To collaboratively serve requests in service groups, each station node runs a fully distributed, utility-driven request dispatching protocol. The protocol allows each station to make local request dispatching decisions through a utility function that jointly optimizes per-view serving cost and balances per-node workload. Its distributed nature achieves desirable scalability and reliability, because it minimizes resource consumption through dispatching and the failure of any node does not hurt the availability of IPTV service.

For the rest of this paper, we focus our discussion on our request dispatching protocol in iCloud. We leave details of service group formation in our technical report [11] due to space limitation.

3 Collaborative Request Dispatching

In iCloud, a user request is first sent to its regional station, which we refer to as the local station of the request. Upon receiving the request, the local node selects the best node within its service group to serve this request in terms of local cache availability and remaining IO bandwidth. Since a user request may be served by multiple candidate nodes, the selection decision should be made to enable most efficient utilization of the shared resources within a service group. We refer to this task as *request dispatching*.

3.1 Design Consideration

Two popular architectures can be applied to address the problem of request dispatching: one is the centralized request dispatching [2, 9] and the other is distributed hash table (DHT) based request dispatching [18]. Both architectures have been successfully deployed in some systems but both have critical limitations as a solution to the request dispatching problem in iCloud.

The centralized dispatching approach sets up a dedicated dispatching server which assigns received requests to a number of nodes based on certain dispatching rules and runtime information collected from nodes. Despite its simplicity, the centralized dispatching may create a single-point-of-failure [9]. In addition, this approach may not scale well with growing number of subscribers and increasing service catalog.

The DHT based dispatching approach, on the other hand, may reduce or avoid single-point-of-failure by mapping an incoming request to a node within a service group based on a distributed hash table. Basically, each node is assigned with a range of keys. If the hashed result of a request falls into the key range of a certain node, the node is assigned with the request. While the DHT based dispatching scheme has been used in several systems, e.g., memcached [6], its static key-based partition does not work well on highly dynamic and skewed workload in IPTV services [7, 15, 19]. As different video content can bring very different workload, the static key range mapping may cause certain servers to be overloaded when they own keys associated with heavy workload. In addition, the per-content workload may change dramatically over time. Even with the support of dynamic remapping of keys, the remapping procedure would have to be invoked frequently to avoid sudden overloading at some nodes, which is not only expensive but also time consuming.

In iCloud, we introduce a fully decentralized request dispatching protocol based on a utility function, aiming at avoiding single-point-of-failure and adapting to dynamic workload at the same time. A key feature of this utility-based request dispatching protocol is that it allows each node make request dispatching decisions individually based on a locally maintained utility function, and guarantees that the overall outcome of distributed request dispatching leads to low per-view serving cost and good load balance among VHO nodes.

3.2 Utility-driven Request Dispatching

When a subscriber submits a request, the request is first sent to its corresponding regional station (local node). The local node then decides which node in its service group should serve the request. There are three possible scenarios: (i) Multiple candidate nodes have the requested content in their local content cache. In this case, a utility function is used to make the decision. (ii) Only one node within the service group has the requested content in its local content cache. This case can be trivially handled by choosing this node to serve the request. (iii) None in the respective service group has the request content in its local cache, which necessitates downloading the content from a remote SHO. After one node downloads the content, it can serve the request as in the previous case. We leave the detail of content downloading in our technical report[11]. In the rest of this section, we describe the design of our utility function, the request dispatching protocol and the performance analysis of our utility-based dispatching protocol.

3.2.1 The Utility Function

With continued advances in modern processors and high speed fibers, data processing and transmission are no longer a performance bottleneck in on-demand IPTV service delivery systems. In contrast, the real bottleneck in these systems is the disk IO. For instance, dense wavelength division multiplexing(DWDM) can easily deliver 320 Gbps video[16]. However, disk access rate has not been improved significantly over time. Since most video servers are disk based, the scalability of on-demand services is fundamentally limited if content is directly read from disk storage. For example, it is difficult to achieve sustained disk data transfer rate of more than 150Mbps per disk driver - corresponding to 15 high definition video stream at a data rate of 10 Mbps. Achieving 100,000 streams would require 6,667 disk drivers, assuming that these disk drivers could be perfectly load balanced, which is not true in reality[16]. Therefore, a common wisdom for scaling on-demand IPTV delivery systems is to utilize memory-based cache to reduce disk IO consumption.

We argue that a good request dispatching protocol should consider its impact on disk IO, because different request dispatching may have different impact on the available disk bandwidth. On one hand, assigning a request r to a node p , which has already been serving requests on the same content requested by r , denoted as $T(r)$, may cause much less disk IO than assigning r to another node q , which is not currently serving any request on content $T(r)$. This is because p may serve r with memory-cached portions of $T(r)$. On the other hand, assigning large number of requests to a node with limited available disk bandwidth may not be a good choice as it may lead to sudden degradation of service quality, e.g., high latency and frequent jitters.

We design the iCloud request dispatching protocol with the following two objectives. First, it should minimize disk IO to reduce the likelihood of disk IO bandwidth being a bottleneck. Second, as services are provided by multiple

nodes, it is essential to maintain consistent service quality over multiple nodes. Because service quality heavily depends on the available disk bandwidth at a node, the second goal indicates that certain levels of load balance in available disk bandwidth should be achieved across nodes. In iCloud we achieve these two goals by enforcing a utility-driven request dispatching protocol at each VHO node, which takes into account of both disk IO bandwidth consumption and disk IO load balance in making the request dispatching decision.

The utility function is designed to consist of two parameters to capture the preference of low IO cost and balanced load. The first parameter is the *bandwidth consumption*, defined as $d(i, j) = \frac{1}{x_{ij}+1}$, where $d(i, j)$ denotes the estimated disk IO of serving content j at node i , x_{ij} is the number of requests on content j currently being served at node i . The rationale is the amount of disk IO caused by a new stream request is inversely proportional to the number of streams being served on the corresponding node, because the more streams being served, the more likely the corresponding portions of the content are cached in memory. Since precise estimation of disk IO for a request is almost impossible, we use $d(i, j)$ to approximate the relative disk IO cost on different nodes. Note that we consider the disk bandwidth consumption for serving any one request directly from disk as 1 for simplicity. It is straightforward to extend it to support heterogeneous cost in terms of disk access.

The second parameter is the *available bandwidth*, denoted by $a(i)$, which is the amount of *available* disk IO bandwidth at node i . Specifically, the value of $a(i)$ can denote the number of requests that can be served directly from disk. In fact, $a(i)$ not only directly models the remaining disk bandwidth, but also models the available streaming cache memory indirectly. For instance, if node i has large amount of available cache memory for streaming, its $a(i)$ tends to be high because it can serve various content directly from its memory cache, which lead the saving of much disk IO bandwidth. However, if node i has little cache memory, it reports small $a(i)$, which makes the other nodes within the same service group to avoid or reduce the number of new requests assigned to node i .

Based on these two parameters, we define the utility function as follows,

$$\phi(i, j) = \frac{d(i, j)}{a(i)} = \frac{1}{(x_{ij} + 1)a(i)} \quad (1)$$

$\phi(i, j)$ measures the portion of disk bandwidth consumption in all available disk bandwidth for node i to serve a request on content j . Hence, $\phi(i, j)$ serves as an indicator of the impact on available resources for serving a request on a node.

Our utility function can easily incorporate other constraints in the request dispatching protocol. For example, we can integrate the network bandwidth constraint in the utility function to avoid assigning requests to the nodes that are congested in terms of available network bandwidth. Specifically, we can extend the definition of $a(i)$ by re-setting $a(i) = \min\{k(i), n(i)\}$, where $k(i)$ is the residual bandwidth of disk components at node i and $n(i)$ is the available

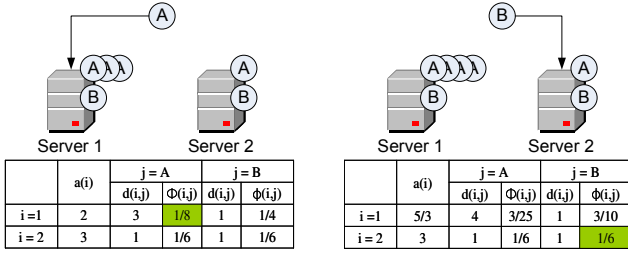


Figure 2. An Example of Request Dispatching

network bandwidth at node i . In general, $a(i)$ can be considered as a general available resource metric, and thus, can be extended to include different resource constraints.

3.2.2 The Dispatching Protocol

The request dispatching protocol in iCloud assigns requests based on the above utility function. Given an incoming request on content j , the dispatching protocol assigns the request to the node i that results in the *smallest* $\phi(i, j)$. By favoring nodes with small $\phi(i, j)$, the dispatching protocol achieves the two goals we listed earlier. On one hand, among nodes with the same available disk IO bandwidth, it prefers those serving the same content and reduces disk IO. On the other hand, when multiple nodes can serve the requested content with similar cost, it favors those that have the most available disk IO bandwidth. Overall, it favors the dispatching that would cause the least impact to the available disk IO bandwidth of nodes. Figure 2 shows an intuitive request dispatching example of two servers. When the first request for content A arrives, the protocol assigns the request to serve 1 as it has a lower utility value (because it is currently serving 3 requests on content A). When the second request for content B arrives, the protocol assigns this request to serve 2 even though both server 1 and 2 are serving the same number of requests on content B. This is because server 2 has more available disk bandwidth, and thus, is more favorable based on the utility function.

Periodically, each node broadcasts its available disk IO bandwidth as well as the set of requests it is currently serving to other nodes within the same service group. We refer to this message as the *workload information message*. Values in this message are used by each node in the service group to update the utility value for every request and node pair based on the utility function. We consider the communication cost of this broadcasting negligible given the capacity of IPTV backbone network. For 100,000 distinct pieces of IPTV content, assuming that each piece of content is at least requested by one subscriber, the size of a broadcast message in the worst case is $16(1 + 100,000) \approx 1.6Mb$. A per-minute-based broadcasting in a service group with 100 nodes consumes about $(2 \times 100 \times 1.6Mb)/60 \approx 5.3Mbps$ network bandwidth at each node, which is trivial given the hundreds of Gbps bandwidth of the backbone network.

In iCloud, each node in a service group maintains a $m \times n$

matrix Φ , where m denotes the number of nodes within the service group and n denotes the number of distinct pieces of IPTV content served or to be served by this service group. At any given time t , the cell in the i th row ($1 \leq i \leq m$) and j th column ($1 \leq j \leq n$) in this $m \times n$ matrix Φ refers to the value of the utility function $\phi(i, j)$ computed at time t . For each node in the service group, upon receiving a workload information from another node i , it updates the i -th row of Φ , which corresponds to content entries related with node i . When a request on content j arrives at node i , the node i performs one round of heap sort over the j -th column of Φ to find the entry with smallest value. For the purpose of efficiency, a node saves the current smallest entry for future incoming requests on content j until a new workload information message arrives. To prevent frequent and insignificant dispatching changes, we introduce a predefined utility threshold Δ_ϕ . A node changes the current dispatching, $j \rightarrow i$ (namely assigning requests for content j to node i) only when $\exists i'$ such that $\phi(i, j) - \phi(i', j) > \Delta_\phi$. This implies that the dispatching will be renewed only when the utility value changes beyond the given threshold compared to the previous utility value.

3.3 Performance Analysis

We have discussed the dispatching protocol of iCloud, where each node in a service group makes dispatching decisions independently without consulting to other nodes. An immediate concern one may raise is that as each node greedily minimizes the utility value when selecting nodes to forwarding the requests it receives, the decisions made at one node could potentially affect the decisions of other nodes. For example, if a node i changes the destination of its requests for content j from node k to node k' , the other nodes in the same service group may also send their requests for content j to k' accordingly due to the reduced serving cost at node k' . Therefore, in this section we present a formal analysis to show that our distributed utility based request dispatching protocol can reach a near-optimal state for all nodes within the same service group.

Our analysis is primarily based on game theory and we borrow concepts and techniques from previous work in game theory[3]. We first define the concept of stable dispatching. Stable dispatching can be informally defined as follows. A set of dispatching is *stable* if no dispatching can strictly decrease its utility value more than Δ_ϕ by changing serving nodes. Formally, we define stable dispatching as follows with the notion Nash equilibrium:

Definition 1 Let \mathcal{G} denote the service group of interest and n denote the number of nodes in \mathcal{G} . A set of dispatching in \mathcal{G} is at Nash equilibrium if for any dispatching $j \rightarrow i$, where j refers to requests for content j and i refers to node i , the following condition holds: $\phi(i, j) - \Delta_\phi \leq \phi(i', j)$, $i \in [1, n] \wedge i \neq i'$. We refer to the set of dispatching that reach Nash equilibrium as stable dispatching.

Based on the definition of stable dispatching, the following theorem shows that the utility-guided request dispatching protocol can always reach a stable dispatching.

Theorem 1 *Given the same incoming request distribution and the same set of nodes, the utility function guided dispatching protocol can always find stable dispatching.*

Proof We first define dispatching potential P as follows,

$$P(m, n) = \sum_{i=1}^m \sum_{j=1}^n \frac{H(x_{ij})}{a(i)}, \quad H(x_{ij}) = \sum_{k=1}^{x_{ij}+1} \frac{1}{k}$$

, where m and n is the number of nodes and the number of distinct content respectively, $H(x_{ij})$ is essentially the $(x_{ij} + 1)$ -th harmonic number. The value of dispatching potential always decreases after each dispatching update. Without loss of generality, we assume that node i shifts its dispatching of request for content j from node p to node q . It leads to $\frac{1}{a(p)x_{pj}} > \frac{1}{a(q)(x_{qj}+1)}$ because otherwise node i has no motivation to change its current dispatching. Thus, the corresponding loss in dispatching potential is $\frac{H(x_{pj})}{a(p)} - \frac{H(x_{pj}-1)}{a(p)} = \frac{1}{a(p)x_{pj}}$, and the corresponding gain in dispatching potential is $\frac{H(x_{qj}+1)}{a(q)} - \frac{H(x_{qj})}{a(q)} = \frac{1}{a(q)(x_{qj}+1)}$. Since $\frac{1}{a(p)x_{pj}} > \frac{1}{a(q)(x_{qj}+1)}$, it is clear the value dispatching potential reduced after node i updating its dispatching for requests on content j . In addition, the reduced value is at least Δ_ϕ , which is not infinitesimal. Since the above assert is true for all nodes on all their re-dispatching, we can safely conclude that the value of dispatching potential always decreases after an dispatching update. Furthermore, because the initial value of dispatching potential is finite, its value only can be reduced by finite times. Therefore, the utility function guided dispatching protocol can always find stable dispatching, given the same incoming request distribution and the same set of nodes. \square

We next show that such a stable assignment can be reached in polynomial time.

Theorem 2 *The time complexity of the utility function guided dispatching protocol is $O(\log x_{max})$, where $x_{max} = \max\{x_{ij} | \forall i, j\}$.*

Proof Let $P_0(m, n)$ and $P_s(m, n)$ to be the initial dispatching potential value and the final dispatching potential value when reaching the stable dispatching. The number of redispaching needed, R , have the following properties: $R \leq \frac{P_0(m, n) - P_s(m, n)}{\Delta_\phi} \leq \frac{P_0(m, n)}{\Delta_\phi} = \frac{1}{\Delta_\phi} \sum_{i=1}^m \sum_{j=1}^n \frac{H(x_{ij})}{a(i)} \leq \frac{1}{\Delta_\phi} \sum_{i=1}^m \sum_{j=1}^n \frac{H(x_{max})}{a(i)} = \frac{nH(x_{max}) \sum_{i=1}^m \frac{1}{a(i)}}{\Delta_\phi} \leq \frac{nm}{a_{min} \Delta_\phi} H(x_{max}) = \frac{nm}{a_{min} \Delta_\phi} \log x_{max}$, where $x_{max} = \max\{x_{i,j} | \forall i, j\}$ and $a_{min} = \min\{a_i | \forall i\}$. Clearly, the number of redispaching required to reach stable dispatching is polynomial to the largest number of requests on a certain content. \square

Stable dispatching essentially means that the set of dispatching made separately by the n nodes in a service group \mathcal{G} can reach a Nash equilibrium[3]. Intuitively, in a stable dispatching no node has anything to gain by changing only its strategy unilaterally. If each node has chosen a dispatching strategy and no node can benefit by changing its

dispatching strategy while the other nodes keep theirs unchanged, then the current set of dispatching choices and the corresponding payoffs constitute a Nash equilibrium.

The above results show that the utility function guided dispatching protocol can reach a stable dispatching in a reasonable time, despite the fact that each node makes greedy local dispatching decisions and different dispatching may influence one another. As the protocol is fully distributed, it avoids the single-point-of-failure, which makes it possible to deliver highly available on-demand IPTV services even in the presence of temporary station failure. Although performance related factors may dynamically change in reality (e.g., the temporal and spatial distribution of user requests) and the setting of control parameters, such as Δ_ϕ , may not be ideal, our experiment results suggest that the iCloud utility-driven request dispatching protocol can achieve significant advantage in service delivery efficiency compared to existing systems with simple dispatching protocol.

4 Experiment Results

We evaluate iCloud through extensive simulation-based experiments over the traces generated based on real-world workload models summarized by previous measurement studies [7, 15, 19]. The simulation process is divided into multiple rounds, each of which represents a fixed length of real-world time. At each round, requests are generated based on variants of Poisson distribution and assigned with requested content, corresponding disk IO rate, watching duration, location of entry, according to statistical distributions contributed by aforementioned studies. The generated requests are sent to nodes in a simulated IPTV service network according to requests' locations of entry. For requests of the same content being served on one node, we assume that the data chunks accessed by these requests follow uniform distribution. Furthermore, if a data chunk has been loaded from disk into memory for a request, then serving another request on the same data chunk does not cause any disk IO.

To understand the performance of iCloud under different settings, we also alter various system characteristics, e.g. service group size, disk bandwidth capacity, and node cache size, and study the consequent effects on the overall performance of the system. Furthermore, we apply different resource constraints in our experiments to model real world deployment scenarios. For instance, we limit the available cache memory for streaming at one node to 5 percent of its local content storage size, and the available network bandwidth at one node to less than 10 times of its disk IO bandwidth. These are common settings used in most of the conventional IPTV systems [16]. To make all the evaluation data statistically meaningful, for each specific configuration, the simulation experiment runs 100 times and hence all data points presented later are averaged over all results. We refer the readers interested in more detail of the workloads and the system models used in our evaluation to our technical report [11].

Through our experimental evaluation, we observe that

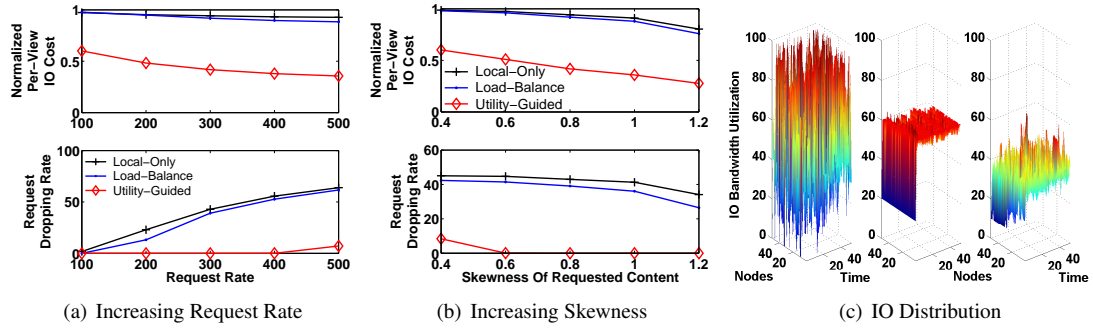


Figure 3. Performance of Request Dispatching Protocols

inter-node collaboration within service group significantly reduces disk bandwidth consumption in request serving. As a result, iCloud provides more than twice the throughput of the existing architecture. We next present detailed results of our experiments in request dispatching.

Figure 4 shows the performance and scalability of different request dispatching protocols. We consider three types of request dispatching in our experiments, *local-only* (local ad-hoc), *load-balance* and *utility-guided*. Existing architecture adopts local-only (local ad-hoc) dispatching, where a node simply assigns an incoming request to itself for serving. With load-balance based dispatching, a node assigns an incoming request to the node with most available disk IO bandwidth within the same service group. The utility-guided dispatching is the protocol used in iCloud, which jointly balances the load and minimizes disk IO bandwidth consumption. We compare the performance of these three dispatching schemes with increasing request rates, increasing request skewness, varying IO distribution, increasing group size and increasing disk IO bandwidth.

Figure 3(a) shows the normalized Per-View (Per-Request) disk IO cost and request dropping rate for all three protocols under increasing request rates. The per-view disk IO cost is normalized by dividing the actual disk bandwidth consumption for serving one request with the disk bandwidth consumption for serving one request without using memory-based cache. In addition, request dropping occurs when a fully-loaded node, i.e., no sufficient available disk bandwidth, is assigned with a request. Clearly, utility-guided dispatching has significantly lower per-view IO cost and lower request dropping rate with increasing request rate, compared with other two protocols. The load-balance based dispatching barely reduces per-view IO cost in comparison to the local ad-hoc dispatching, and it reduces request dropping rate given small workload, as it utilizes lightly loaded nodes.

Figure 3(b) shows the performance comparison of the three protocols with increasing skewness, i.e., the skewness factor in Zipf distribution. Again we measure the per-view IO cost and the request dropping rate. Regarding Zipf distribution, the probability of the rank k element is $P_k = \frac{1/k^s}{\sum_{i=1}^N m_i^s}$, where s is the skewness factor. The larger s is, the higher probability elements with small ranks (pop-

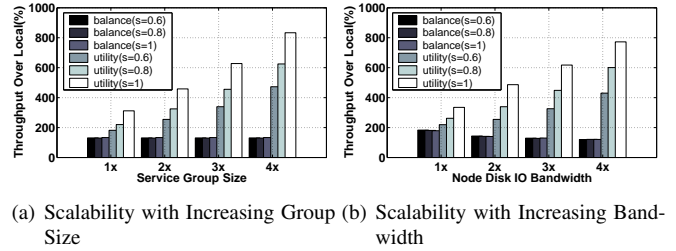


Figure 4. Scalability of Request Dispatching Protocols

ular content) has. Utility-guided dispatching also benefits more from more skewed distribution, since more requests ask for a few popular content and they can be served more efficiently with memory-based caches.

To provide a better understanding on the performance of different protocols, we use Figure 3(c) to show the distribution of disk bandwidth utilization of nodes over a period of time under different protocols. The left plot shows the case for local-only dispatching. Since the surface filled with highly different utilization, it suggests that local-only suffers from dynamics in both location and time of user requests. The middle plot presents the utilization distribution for load-balance dispatching, which provides a much more smooth surface. The right plot depicts the case for utility-guided dispatching, where we see a less smooth surface (compared with load-balance), but much lower average IO bandwidth utilization (consumption).

Figure 4(a) and 4(b) illustrate the scalability of load-balance and utility-guided dispatching under increasing service group size and per-node disk bandwidth respectively. Note that we measure the relative improvement of throughput produced by these two protocols over that of local-only dispatching. In addition, recorded throughput is the maximum overall request rate with less than 10% request dropping. From both figures, we observe remarkable advantage in utility-guided dispatching compared with load-balance based dispatching. Utility-guided dispatching yields higher relative throughput given increasingly more resources. Furthermore, the utility-guided dispatching reacts more favorably with more skewed requested content distribution. An-

other interesting observation is that the relative throughput improvement of load-balance based dispatching drops with the increasing per-node disk bandwidth in Figure 4(b). The reason is due to the fact that there is decreasing dynamics of node utilization for local-only dispatching with increasing disk IO bandwidth.

5 Related Work

Much previous work studied PC-based P2P IPTV services[8, 10, 17]. These works are not closely relevant to ours as we study commercial IPTV services with dedicated infrastructure. In addition, Allen [2] discussed the possibility of deploying efficient VoD services over traditional cable networks with P2P proxy cache on set-top boxes, while our work studies collaboration among stations in IPTV network and avoids modifying users' set-top boxes and related privacy issues.

Several previous work addressed deployment and network design of commercial IPTV and streaming media systems[1, 5]. Agrawal et al.[1] developed a methodology to aid service providers to effectively plan the deployment of IPTV services to maximize the return-on-investment. Cherkasova[5] proposed a capacity planning tool for streaming media services. Compared with clean-slate approaches proposed in these works, we aim at scaling and improving the deployed network of IPTV stations with overlay middleware.

The architecture of iCloud shares some similarities with Content Delivery Network(CDN)[13, 4]. However, compared with previous CDN work, our work makes several unique contributions. First, we studied request dispatching from the perspective of disk IO consumption as disk IO bandwidth is usually the performance bottleneck at the service station side. Second, iCloud employs a fully distributed dispatching protocol which is designed based on best response strategies[3] in game theory. Third, we systematically studied the performance of best response strategies in the request dispatching problem. We speculate that iCloud may also be complementary to existing CDN work.

6 Conclusion and Future Work

This paper presents iCloud, an IPTV overlay framework for scaling on-demand IPTV services. We identified the inherent problems with the conventional one-to-many mapping architecture between IPTV regional stations and subscribers. We proposed iCloud to organize IPTV regional station nodes into service groups, and encourage nodes to collaboratively serve requests based on a utility function. Our theoretical and empirical results show that iCloud significantly improves the performance and scalability of on-demand IPTV services. To the best of our knowledge, iCloud is the first work that adapts existing IPTV service architecture to support inter-station collaboration. As a part of our on going work, we are studying the architectural and algorithmic support for reliable high definition streaming from multiple IPTV sources in iCloud.

Acknowledgment

The first two authors are partially supported by grants from NSF ISE NetSE program, CyberTrust program, an IBM faculty award, IBM SUR grant and a grant from Intel Research Council. The third author is partially supported by National Natural Science Foundation of China under grant No. 60703042 and National Key Science and Technology Research Program of China under grant 2009ZX01043-003-003.

References

- [1] D. Agrawal, M. Beigi, C. Bisdikian, and K.-W. Lee. Planning and managing the iptv service deployment. In *Integrated Network Management*, pages 353–362. IEEE, 2007.
- [2] M. S. Allen, B. Y. Zhao, and R. Wolski. Deploying video-on-demand services on cable networks. In *ICDCS*, 2007.
- [3] E. Anshelevich, A. Dasgupta, J. M. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *FOCS*, 2004.
- [4] J. G. Apostolopoulos, T. Wong, S. J. Wee, and D. Tan. On multiple description streaming with content delivery networks. In *INFOCOM*, 2002.
- [5] L. Cherkasova, W. Tang, and S. Singhal. An sla-oriented capacity planning tool for streaming media services. In *DSN*, pages 743–752, 2004.
- [6] B. Fitzpatrick. Distributed caching with memcached. *Linux Journal*, 2004(124):5, 2004.
- [7] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, November 2006.
- [8] M.-T. Lu, H. Nien, J.-C. Wu, K.-J. Peng, P. Huang, J. J. Yao, C.-C. Lai, and H. H. Chen. A scalable peer-to-peer iptv system. In *CCNC*, pages 313–317, 2007.
- [9] D. Lui. Distributed architecture vs centralized architecture for ip vod. *Annual Review of Communications*.
- [10] J.-G. Luo, Y. Tang, M. Zhang, L. Zhao, and S.-Q. Yang. Design and deployment of a peer-to-peer based iptv system over global internet. In *ChinaCom*, pages 1–5, 2006.
- [11] S. Meng and L. Liu. A service overlay for scaling on-demand iptv services. Technical report.
- [12] C. Moulynox. Iptv bandwidth growth. <http://www.webtvwire.com/iptv-bandwidth-growth-60-of-all-consumer-ip-traffic-in-2008/>.
- [13] S. Saroiu, P. K. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. In *OSDI*, 2002.
- [14] K. Scherf, H. Wang, and Y. Jiang. Iptv: From quadruple play to multiplay. *Industry Report, PARKS ASSOCIATES*, 2007.
- [15] T. Silverston and O. Fourmaux. Measuring p2p iptv systems. In *NOSSDAV*, 2007.
- [16] SUN. Redefining iptv with the sun streaming system scalable video-on-demand and network personal video recording for an open systems world. White Paper 08.
- [17] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. Measurement of a large-scale overlay for multimedia streaming. In *HPDC*, pages 241–242, 2007.
- [18] X. Wang, W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Buddyweb: A p2p-based collaborative web caching system. In *NETWORKING Workshops*, 2002.
- [19] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *EuroSys*, 2006.