The Feasibility of DHT-based Streaming Multicast

Stefan Birrer Fabián E. Bustamante
Department of Computer Science
Northwestern University, Evanston, IL 60201, USA
Email: {sbirrer,fabianb}@cs.northwestern.edu

Abstract

We explore the feasibility of streaming applications over DHT-based substrates. In particular, we focus our study on the implications of bandwidth heterogeneity and transiency, both characteristic of these systems' target environment. Our discussion is grounded on an initial evaluation of SplitStream, a representative DHT-based cooperative multicast system.

1. Introduction

The limited deployment of IP Multicast [19, 20] has led to considerable interest in alternate approaches implemented at the application layer, relying exclusively on end-systems [23, 16, 2, 15, 30, 8, 13]. Among the proposed end-system multicast protocols, tree-based systems have proven to be highly scalable and efficient in terms of physical link stress, state and control overhead, and end-to-end latency.

Conventional tree-based structures, however, are inherently not well matched to the characteristics of cooperative distributed environments. Cooperative settings, in which participating peers contribute resources in exchange for some service, have been found to be highly dynamic and heterogeneous in terms of node resource availability and uptimes [35, 36, 10, 29]. Tree-based multicast structures are problematically highly dependent on the reliability of non-leaf nodes and are likely to be bandwidth constrained, with bandwidth availability monotonically decreasing as one ascends from the leaves. In addition, in tree-based multicast systems the burden of data forwarding is carried by a small fraction of non-leaf nodes, clearly conflicting with the expectations of a cooperative environment [13].

A number of recently proposed protocols [30, 24, 13, 9] explicitly address these issues by distributing the forwarding load among all participants, thus lowering system dependency on any particular node. While the proposed techniques can be equally applied to both performance-based [17, 23, 2, 30, 8] and DHT-based systems [34, 37, 31], the latter path is particularly compelling as the resulting DHT substrate could potentially be used by multiple applications, consequently allowing each to leverage the manifold advantages of this approach while sharing the total control overhead.

It is therefore of interest to explore the feasibility of cooperative multicast applications over DHT-based overlays. This paper presents such an analysis focusing, in particular, on the requirements of media streaming multicast applications and the implications of the high degree of transiency and widely heterogeneous bandwidth availability among participating nodes in these systems' target environments [5, 36]. A closely related work by Bharambe et al. [5] studies the impact of heterogeneous bandwidth limitation, focusing on the implications of such constraints on a representative DHT-based multicast protocol, Scribe [15]. Our work complements and extends their effort by also considering the implications of high degrees of churn in peer populations in the context of cooperative DHT-based protocols. While our analysis is grounded on an evaluation of SplitStream [13], a representative and relatively mature cooperative DHT-based system, we believe most of our conclusions and some of our proposed solutions apply well to other DHT-based protocols.

Our study was prompted by our wide-area experimentation with this class of systems [6]. For the purpose of this analysis, however, we resort to simulations as this allows us to observe the dynamic behavior of these protocols in controlled and repeatable settings. Our discussion is structured around the aforementioned issues and the performance demands of this class of applications – bandwidth-intensive streaming multicast. Our evaluation indicates that SplitStream's performance and its ability to handle transiency and honor bandwidth constraints appear to be limited by its design choice to build on a DHT substrate. Throughout the paper, we discuss possible ways to address some of the identified problems in DHT-based multicast.

We begin by providing some necessary background on the evaluation framework in Section 2 and the evaluated protocols in Section 3. In Sections 4 and 5, we describe the experimental setup and present our analysis of the feasibility of DHT-based, bandwidth-demanding, streaming applications. We present related work in Section 6, discuss our £ndings in Section 7 and conclude in Section 8.

2. Evaluation Framework

Our evaluation is based on the requirements of streaming multicast applications, where one or many sources stream audio/video data, at several hundred kilobits per seconds, to a potentially large set of heterogeneous receivers with small, bounded delivery latencies [16].

We analyze the implications of the high degree of transiency and the wide range of resource availability among peer populations in these systems' target environments. Subscribers in such settings may join and leave the session at any given time, or even fail. Median session times have been reported to range from a few hours down to, more commonly, a few minutes [35, 10, 22, 16, 36]. Highly transient peer populations force protocols to constantly probe and reevaluate peers and, if needed, repair their distribution topology. Control messages associated with such tasks could easily consume a substantial fraction of the peers' resources. Peer resource availability, including bandwidth capacity, have been found to differ by over four orders of magnitude [35, 36, 5, 29]. Adequate placing of nodes in the hierarchy based on available bandwidth is a non-trivial task, additionally complicated by the transiency of the population and the dynamic nature of bandwidth availability.

Addressing such degrees of transiency and heterogeneity without sacri£cing application performance is a critical challenge for the end-system multicast approach in general, and media streaming applications in particular.

3. Background

In order to set the stage for the following discussion, this section provides an overview of SplitStream and the underlying protocols upon which it builds.

3.1. SplitStream

Addressing the unsuitability of conventional tree-based protocols to cooperative environments, Castro et al. [13] propose to split the multicast content into k stripes and to multicast each stripe using a separate multicast tree, where an interior node in one tree is a leaf node in all others. By more evenly distributing the forwarding load among participating peers, this approach avoids the problem of underprovisioned interior nodes. Additionally, it increases the robustness of the system to node failures as the disjoint trees reduce the dependency on any single node. In an attempt to control inbound bandwidth consumption, a peer joins at most as many stripes as its bandwidth capacity permits. In [13], the authors present two alternative policies for constructing SplitStream's multiple trees – one variant that emphasizes fairness, and a second that trades perfect fairness for improved performance. The performance-optimized variant uses outdegrees limited to the available bandwidth capacity. The SplitStream implementation by Castro et al. relies on Scribe [15], an application-level group communication protocol built on Pastry [34], a proximity-aware, structured peer-to-peer routing protocol.

3.2. Pastry

Every peer in Pastry [34] is assigned a randomly unique 128-bit node identi£er (nodeId). NodeIds are thus uniformly distributed in the circular identi£er space formed by all possible identi£ers. A nodeId can be expressed as a sequence of digits in base 2^b, where b is a con£guration parameter with a typical value of 3 or 4. Given a message and an associated 128-bit key, Pastry routes the messages to the node with the numerically closest nodeId. In order to route messages, each node maintains a routing table consisting of $\log_{2^b} N$ rows and 2^b columns, where the node associated with each entry in row r of the routing table shares the £rst r digits with the local node's nodeId. A message is routed to a node whose nodeId shares a pre£x with the message key of at least one digit longer than the current node's nodeId. If no such node exists, the message is routed to a node whose nodeId shares a pre£x with the message identi£er that is as long as the current node's nodeId, but is numerically closer to the key. Additionally, each node maintains a leaf set and a set of neighboring nodes. The leaf set contains n nodes which are numerically closest to the local node's nodeId ($l/2$ with smaller identi£ers and $l/2$ with larger ones), whereas the neighborhood set consists of nodes which are closest based on a proximity metric. While the neighborhood set is not normally used for routing, the leaf set is used whenever the destination key falls within the space of the leaf set, to £nd a node with a numerically closer nodeId. In order to provide routing through the network, the Pastry overlay requires a consistent mapping from keys to overlay nodes and depends on persistent intermediate nodes for successful message delivery.

Routing application message through a DHT (**DHT-based routing**) yields a number of bene£ts. DHTs provide robust routing in the presence of link failures. As long as there exists a path from source to destination through intermediate nodes, a message can be delivered even though the direct Internet connection may be temporarily down, perhaps as a side effect of BGP route conversions. In addition, routing through a DHT may allow an application to take advantage of faster available paths, as routing through an intermediate node may expose a faster end-to-end path than the direct route, e.g., as a result of BGP policies for inter-AS routing. Routing through intermediate nodes, on the other hand, increases the system load in term of bandwidth utilization and may also add extra processing latency to each forwarded message. Caching of destination IP addresses would enable an application to route directly without going through the DHT, i.e., routing using **non-DHT links**, potentially resulting in lower delivery latencies and lower overall bandwidth consumption at the end hosts. Direct routing to the destination, however, may add network links that are not part of the underlying DHT, thus giving up on some of the bene£ts of robust routing, including node and path failure detection among others.

3.3. Scribe

The Scribe [15] group communication builds upon Pastry to support applications that demand large number of multicast groups. Each of these multicast groups may consist of a subset of all nodes in the Pastry network. Every multicast group in Scribe is assigned a random ID (topicId), and the multicast tree for the group is formed by the union of Pastry routes from each group member to the root, identi£ed by the topicId. Messages are then multicast from the root using reverse path forwarding [18]. Since Scribe relies on the underlying DHT substrate routing table for data forwarding, it naturally leverages the robustness, self-organization, locality and reliability properties of Pastry. As any DHT-based application, Scribe may opt for routing messages through the DHT or over non-DHT links maintained by the Scribe layer.

Scribe also enables higher level protocols to specify policies that can be used, for example, to enforce outdegree requirements [13, 5]. This makes it possible to limit the nodes' outdegree, i.e. the number of successors in the distribution topology, by letting applications deny join requests. Alternatively, in order to increase the general ef£ciency of the distribution tree, rather than denying a new request, a protocol may choose to preempt a current successor. In heterogeneous environments, this enables pushing bandwidth capable nodes up the tree, thus increasing the overall system performance.

4. Evaluation

We study the feasibility of DHT-based streaming multicast by analyzing the behavior of a representative system based on a few key metrics. The performance of streaming multicast applications is evaluated in terms of delivery latency and delivery ratio. **Delivery Latency** is de£ned as the end-to-end delay (including retransmission time) from the source to the receivers, as seen by the application. This accounts for path latencies along the overlay hops, as well as queuing delay and processing overhead at peers along the path. **Delivery Ratio** is the ratio of subscribers that have received a packet within a £xed time window. Disabled receivers are not accounted for.

The effectiveness of distributing the load among all participants is analyzed in terms of physical outdegree. **Physical Outdegree** is de£ned as the fanout of a node and indicates the number of full rate streams a peer has to support in the distribution topology. Whenever multicast data is split into smaller messages and peers only forward part of them (e.g., SplitStream), the effective outdegree can be a non-integer value.

The remainder of this section describes our experimental setup and presents implementation details of the compared protocols.

4.1. Details on Protocol Implementations

For our evaluation, we rely on the implementation of the protocols used for the same purpose in their original publications – for both Scribe [15] and SplitStream [13] we employ FreePastry (version 1.3.2) [21]. To this end, we ported FreePastry's communication layer to our Reef middleware framework [7] for simulation.

Scribe and SplitStream use a leaf set maintenance interval of 60 seconds and a route set maintenance interval of 900 seconds [13]. The outdegree for SplitStream nodes is limited to the number of stripes for the fair variant [13]. We evaluate SplitStream with 16 stripes (trees), where each of the stripes is responsible for forwarding $\frac{1}{16}$ of the full data rate to each client. Thus, an outdegree of one in a SplitStream tree corresponds to a physical outdegree of $\frac{1}{16}$. Setting SplitStream's outdegree to 16 yields an effective utilization of one full-rate stream at each node, i.e., a physical outdegree of one.

In the presence of failures, degraded performance may result from the time needed to detect and repair failed nodes in the distribution topology. Moreover, repairing the routing structure results in additional network traf£c which, in turn, may lead to congestion collapse [33, 11]. In order to improve the generality of our conclusions, we factor out the failure detection component, by allowing the evaluated protocols to detect failed nodes virtually instantaneously, as well as the effect of control message overhead, by not modeling bandwidth in our simulation. Given this considerations, we believe the reported £ndings are generalizable to other Pastry implementations [33, 11] and, to a large extent, to other DHT substrates as well.

4.2. Experimental Setup

We ported FreePastry's communication layer to the Reef framework. Reef [7] is a middleware framework that supports the full cycle of experimental research in overlay systems – from design, through implementation, to evaluation and analysis. Researches can specify their algorithms in a well-known object oriented language, Java, leveraging a rich set of libraries that are part of Reef. For this purpose, Reef provides a generic interface for access to network services such as sending and receiving of messages, as well as for querying round-trip times and loss rates among other properties. Reef's protocol implementations run in simulation and wide-area settings, such as PlanetLab, without changes to the code base. Reef shortens the development cycle by removing the burden of implementing commonly used services and by reducing the complexity of maintaining separate implementations for simulation and wide-area experimentations. For simulation, Reef relies on SPANS, a packet-level, event-based simulator. SPANS supports several common topology £le formats as well as its own extensible scripting language. The network is modeled as a set of nodes connected by links, each having a bandwidth capacity, latency and loss rate. Each of the nodes can be a router or an end host. An end host differs from a router in that it has agents (running clients) attached to it. Messages are routed through the graph built by the nodes and links using Dijkstra's shortest path algorithm. Network queues at the routers use tail-drop buffers to emulate the loss and delay properties arising from the links' limited bandwidth capacity.

We ran our simulations using Brite [28] topologies with 2,000 nodes and a multicast group of 256 members. Latency was set according to the node's/router's physical distance; bandwidth and loss rate were not modeled. While measuring the nodes' outdegree, we chose not to model bandwidth and loss rate, to avoid the negative effects of control traf£c. Each simulation experiment lasts for 60 minutes of simulation time, including a warm-up time of 57 minutes, during which the protocols establish and optimize their structures. All nodes join the network at times distributed over the £rst 30 minutes. Scribe and SplitStream are activated after Pastry runs for 10 minutes [13]. Following the warm-up period, each simulation has a 180-second phase with rapid membership changes. During this time each protocol is exercised under different failure rates sampled from an exponential distribution. We vary the *mean time to failure (MTTF)* from 1 minute to 2 hours; the *mean time to repair (MTTR)* is £xed at 10 minutes. A session with MTTF of 5 minutes models an environment where nearly 60% of the peer population fails during the three minute failure interval. All experiments were run with a payload of 1000 bytes. We employed a rate of 10 packets per second, which corresponds to a data rate of 80 Kbps, a realistic scenario for applications such as multimedia streaming.

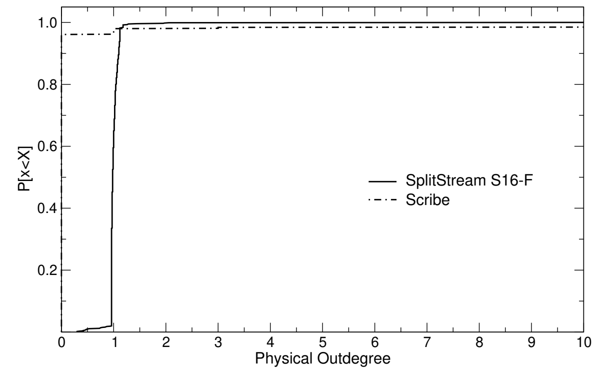


Figure 1: Physical Outdegree CDF (256 end hosts, no failures). Physical outdegree de£nes the bandwidth contribution in terms of full-rate streams at each node. Since the stream is split into multiple packets and each of this packet is forwarded in the distribution topology following inherent forwarding rules, the nodes' effective bandwidth consumption depends on the employed forwarding rules and on the nodes' position(s) within the distribution topology. Note that an outdegree of one in SplitStream corresponds to a physical outdegree of $\frac{1}{16}$.

5. Analysis

We structure our analysis around the two identi£ed challenges for overlay streaming multicast – all overlay multicast systems must cope with the higher failure rate of end nodes when compared to routers, while respecting the nodes' bandwidth constraints, all without sacri£cing application performance.

5.1. Bandwidth Constraints

The scalability of an overlay multicast protocol is in part determined by the forwarding responsibility of each participating peer. Figure 1 shows the Cumulative Distribution Function (CDF) of physical outdegree for the fair variant of SplitStream and Scribe. The physical outdegree is the forwarding capacity used at each node in terms of a basic stream rate, i.e., a physical outdegree of one indicates that the peer contributes exactly the equivalent of one full rate stream to the system. We see that this variant of SplitStream shares the forwarding load evenly among the participating peers with most of them having a physical outdegree of one. In homogeneous environments, such as some corporate settings [11], this improves on the scalability of conventional tree-based multicast, such as Scribe, where

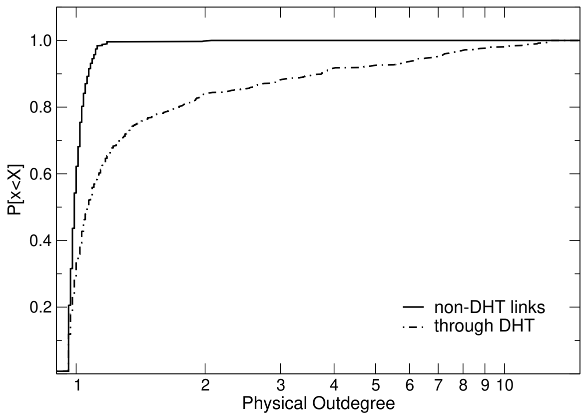


Figure 2: Physical Outdegree CDF (SplitStream, 256 end hosts, no failures). This graph shows the physical outdegree of the nodes when routing all data messages through the DHT substrate and when using non-DHT links for routing. Note that an outdegree of one in SplitStream corresponds to a physical outdegree of $\frac{1}{16}$.

only a subset of the nodes, i.e., the interior nodes, share the total forwarding load.

In heterogeneous environments, nodes may have different inbound and outbound bandwidth constraints. In order to control their inbound bandwidth consumption, nodes participating in a SplitStream multicast tree may subscribe to a subset of the available stripes. Limiting the number of stripes a node is subscribed to, however, will also reduce the robustness gained through the usage of multiple distribution trees. Additionally, as DHT-based systems may require a node to subscribe to a particular stripe de£ned by their node pre£x [13], employed encoding schemes are potentially constrained.

The outbound bandwidth consumption of a SplitStream/Scribe node can be controlled by limiting the number of successors in the distribution tree. Honoring outgoing bandwidth constraints, however, may force these protocols to utilize overlay links that are not part of the underlying DHT. In fact, previous studies have found that non-DHT links make over 40% of all links used by Scribe in bandwidth limited homogeneous environments, and over 80% in heterogeneous environments [5].

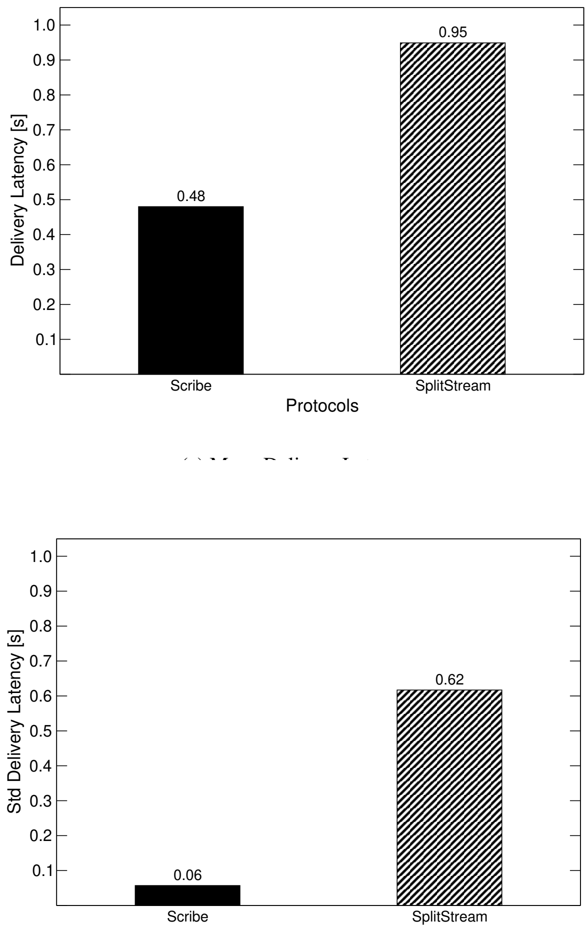
Using these non-DHT links requires the protocol to use an independent algorithm on top of the DHT substrate to maintain these links, monitor dynamic bandwidth availability, and detect node failures, thus incurring additional control overhead. This additional control traf£c, which is particular to an application, negates some of the bene£ts of the DHT-based approach, speci£cally the sharing of control overhead across multiple applications.

Routing through the DHT, on the other hand, adds extra forwarding load on intermediate nodes, thus reducing the scalability of the protocol. Figure 2 shows the CDF of the physical outdegree for overlay nodes when all messages are being routed through the DHT substrate or over non-DHT links. Routing through the DHT adds extra forwarding load on intermediate nodes and results in 10% of the nodes having an outdegree larger than 6. Higher layer protocols are not aware of this additional forwarding load introduced by DHT-based routing, as this traf£c is routed in the underlying DHT substrate without involving the application.

Heterogenous environments may also result in deep Scribe distribution trees, in part because of the disconnect between bandwidth capacity and nodeId assignment [5]. In general, a signi£cantly longer distribution path may result in higher performance loss. While assigning nodeIds based on the node's bandwidth constraints may help reduce the median tree depth [5], it can also adversely affect the routing properties of DHTs [4]. The key-based routing of DHT substrates con¤icts with the performance optimization goals of multicast schemes, such as maximizing available bandwidth and minimizing end-to-end latency. Unfavorably, current systems account for identi£er pre£x at £rst and for proximity to break ties at second when building the routing tables [34].

SplitStream achieves a more even distribution of forwarding load among participating peers by splitting the multicast content into k stripes and multicasting each stripe using a separate multicast tree. Each of these trees uses a Scribe distribution topology rooted at a randomly located node that is de£ned by the topicId. As messages are multicast from the root using reverse path forwarding, publishing incurs an additional source-to-root path delay penalty. Since the root node of the multicast tree is selected based solely on the node's nodeId, an overloaded and/or bandwidth-limited node could potentially reduce the overall group performance. Additionally, using multiple interior-node-disjoint trees leads to only suboptimal solutions in terms of end-to-end latency, since a node's parent can only be recruited from a subset of all participating peers, emphasizing fairness over performance. Figure 3 (a) illustrates the impact of using multiple interior-node-disjoint trees. SplitStream using 16 stripes increases the delivery latency by 98% compared to Scribe. We expect this effect to be less pronounced with very large groups, due to the higher density of participating peers.

A 16-stripe SplitStream multicast session utilizes 16 Scribe multicast trees. Each of these trees has been assigned uniformly distributed identi£ers and, for each Scribe tree, the node with the closest node identi£er will serve as the root of the corresponding multicast tree. A client subscribing to the SplitStream session will receive messages from all the geographically distributed sources together and



(b) Std Delivery Latency.

Figure 3: Delivery Latency (256 end hosts, no failures). SplitStream uses 16 strips to disseminate the multicast data.

will consequently incur an extra variance de£ned by the geographical distribution of these root nodes. This variance is larger than when the messages are streamed from a single source, as with Scribe. Thus, SplitStream's delivery latency variance is mainly de£ned by the mean message delivery latency difference among the various root nodes and not by the variance of message delivery latency within one of its trees. Figure 3 (b) illustrates this effect well. The £gure shows the standard deviation of delivery latency for both Scribe and SplitStream with 16 stripes. SplitStream shows 900% increase in the standard deviation of message delivery latency with respect to Scribe.

5.2. Transiency

Measurement studies of widely used peer-to-peer systems have reported median session times, i.e. the time from the node's joining to its subsequent leaving the system, ranging from two hours [35] down to a minute [10, 22, 16, 36]. Ef£ciently handling this high level of transiency, while still delivering good performance to the application, has proven to be a dif£cult task [33, 16, 27, 11, 26, 32]. While Pastry has been successfully applied to less transient environments [38], early Pastry implementations [21] in particular have shown to perform poorly under medium to high churn with a median session time less than 20 minutes [33]. Newer protocols, such as MSPastry [11] and Bamboo [33], have been designed to avoid congestion collapse under high transiency. To avoid congestion collapse, however, these protocols trade performance for less control overhead, resulting in an increased relative delay penalty under high churn. Speci£cally, the relative delay penalty for messages under transiency with a median session time of 5 minutes is more than 100% higher than with a median session time of 2 hours [11]. While some applications may not be sensitive to this extra delay, streaming multicast application require low delivery latency in order to timely detect and recover missing packets.

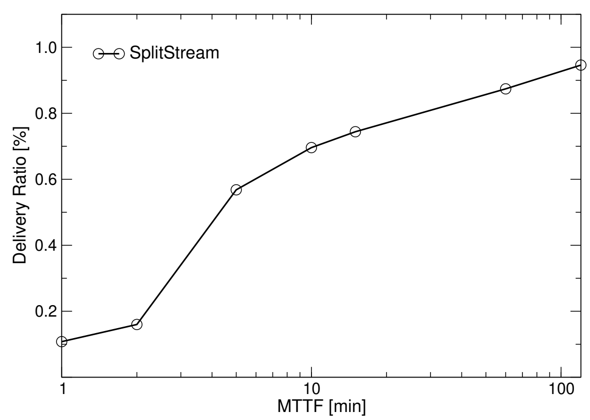


Figure 4: Transiency (SplitStream, 256 end hosts). Note that the MTTR was £xed at 10 minutes for the different failure rates.

Churn may negatively affects the ability of a multicast protocol to deliver application messages. Figure 4 illustrates SplitStream's robustness to node failures with different degrees of transiency in terms of delivery ratio. Note that SplitStream delivers about 95% percent of the packets with a median session time of 2 hours, a level of transiency expected from corporate settings. With higher failure rates, however, the delivery ratio signi£cantly declines. In particular, the delivery ratio for median session times ranging from 5-10 minutes, a degree of churn representative for streaming multicast [36], lies between 50% and 75%. It is possible that DHT resilience could be improved by adopting techniques from performance-based resilient multicast systems [3, 40, 8].

Large routing tables help reduce the total end-to-end latency, thus DHT substrates generally prefer routing tables with tens to, more commonly, hundreds of neighbors. High degree of transiency, on the other hand, may result in a storm of routing table updates with their associated control overhead [25, 11], presenting a problem especially for large routing tables. Since reacting to each membership update could result in congestion collapse, in order to reduce bandwidth consumption one could opt for ignoring routing table updates. This, however, may degrade the application performance as the distribution topology diverges from the current optimal tree according to the routing tables. It seems that protocols with smaller average number of neighbors provide higher resilience to churn than those with larger routing tables.

A mostly ignored aspect of highly transient environments is the setup time for joining peers. As a large number of nodes join a session over a relatively short time period, it is possible that techniques designed to control the impact of high transiency will have a negative effect on the quality of service perceived by new peers. Previous studies found the median join time for nodes in a Pastry system with high transiency (5 minutes median session time) larger than 10 seconds [11]. This median join or setup time could be improved through a hierarchical approach, where peers with longer expected lifetime form a more stable core network to which newcomers could join [10]. Of course, the adoption of a hierarchical scheme may come at a cost on application performance, as more stable routes are chosen over newer, more desirable ones.

6. Related Work

Peer-to-Peer multicast streaming applications have recently drawn signi£cant attention. Chu et al. [16] report on their experience deploying a video broadcast service for various events, such as live conference broadcast. Sripanidkulchai et al. [36] analyze the feasibility of supporting large-scale groups using application end-point architecture and report that, in most scenarios, end hosts have suf£cient resources to support peer-to-peer multicast. In a closely related study, Bharambe et al. [5] analyze the impact of heterogeneous bandwidth constraints on DHT-based multicast protocols and conclude that Scribe tends to create deep unbalanced distribution trees under these conditions. Our work extends this study by analyzing the implications of transiency and bandwidth heterogeneity on the effectiveness of DHT-based, cooperative multicast approaches.

The performance of DHT-based multicast systems has been the focus of several studies. Banerjee et al. [1] describe and analytically compare a set of proposed application layer protocols including DHT-based and tree-based techniques. Castro et al. [14] contrast CAN-style versus Pastry-style overlay networks using multicast communication workloads, and conclude that multicast trees built on Pastry provide higher performance than those using CAN [31]. Focusing on data-sharing applications, Castro et al. [12] compare structured and unstructured Gnutella-like peer-to-peer systems. Our work analyzes the feasibility of DHT-based systems in the well-de£ned context of streaming multicast in transient, heterogeneous environments.

Various efforts have been proposed to increase the resilience of DHT substrates. Li et al. [25] propose to dynamically adapt the routing tables based on the system's current degree of transiency. Castro et al. [11] propose bandwidth-ef£cient algorithms to reduce the bandwidth used to maintain and repair the nodes' routing tables. Furthermore, Bamboo [33] introduces a periodical recovery scheme to increase the DHT's ability to handle churn.

7. Discussion

We have structured our analysis around the two identi£ed challenges for overlay streaming multicast – the high degree of transiency and wide range of resource availability among peer populations in large-scale, open settings. Our analysis indicates that such degrees of transiency and heterogeneity are challenging for DHT-based streaming multicast systems. Most proposed solutions in the literature addressed some of these issues at the cost of application performance.

DHT substrates employ unique node identi£ers and build their routing structures based on random node identi£ers and a latency metric to break ties. The high-throughput and low-latency requirements of streaming multicast applications, however, are not well re¤ected in this policy. While support for bandwidth capacity has been built into DHT-based applications, it is unclear how such schemes may effectively deal with the dynamic nature of the link's available bandwidth capacity. A number of heterogeneity-aware techniques for DHT-based multicast protocols have been proposed in the literature [5, 39]. To effectively honor bandwidth constraints, the proposed approaches introduce non-DHT links which, not being supported at the underlying DHT substrate, land in the realm of the applications who are now responsible for monitoring these links, as well as detecting and repairing failures.

Although we have conducted this study with medium sized peer groups, we acknowledge that using larger populations may potentially bene£t other DHT-based applications as DHTs were initially designed with large scales in mind. For instance, SplitStream has been shown to scale well in terms of latency and fairness with increased group sizes [13] in corporate settings. In the context of streaming multicast over dynamic, heterogeneous networks, however, we do not expect larger scales to help addressing the identi£ed issues with DHT-based multicast.

Some of the issues we have raised could potentially be addressed through the adoption of ideas from performance-based systems. It is, however, an open question whether those techniques can be incorporated into DHT-based systems without con¤icting with some of these systems' basic concepts.

8. Conclusions

We analyzed the impact of highly dynamic and heterogeneous environments on DHT-based streaming multicast. Our discussion is based on a detailed evaluation of SplitStream [13] and the two underlying protocols it builds on, Scribe [15] and Pastry [34]. While these underlying protocols have been successfully applied to homogeneous and stable settings, such as corporate environments, the characteristics of more open networks have proven to be a challenge.

Layering protocols enables researchers to more easily compose powerful applications. However, careful attention should be paid to the £tness of the resulting "stack" for a particular purpose. Compromises made at the lower levels of the stack with one class of applications in mind may result in a weak platform for a different class of applications built on top of it. While splitting a stream content into multiple stripes, forwarded over a multicast forest, is a powerful concept for building streaming multicast systems, its realization over a DHT-based substrate will in part depend on effectively addressing some of their issues pointed out in this paper.

References

[1] S. Banerjee and B. Bhattacharjee. A comparative study of application layer multicast protocols, 2002. Submitted for review.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. of ACM SIGCOMM*, August 2002.

[3] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *Proc. of ACM SIGMETRICS*, June 2003.

[4] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting scalable multi-attribute range queries. In *Proc. of ACM SIGCOMM*, August/September 2004.

[5] A. R. Bharambe, S. G. Rao, V. N. Padmanabhan, S. Seshan, and H. Zhang. The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols. In *Proc. of IPTPS*, February 2005.

[6] S. Birrer and F. E. Bustamante. The costs of resilience in overlay multicast protocols. Tech. Report NWU-CS-04-50, Northwestern U., October 2004.

[7] S. Birrer and F. E. Bustamante. Reef: Ef£ciently designing and evaluating overlay algorithms. Tech. Report NWU-CS-05-14, Northwestern U., July 2005.

[8] S. Birrer and F. E. Bustamante. Resilient peer-to-peer multicast without the cost. In *Proc. of MMCN*, January 2005.

[9] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda. FatNemo: Building a resilient multi-source multicast fattree. In *Proc. of IWCW*, October 2004.

[10] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *Proc. of IWCW*, October 2003.

[11] M. Castro, M. Costa, and A. Rowstron. Performance and dependability of structured peer-to-peer overlays. In *International Conference on Dependable Systems and Networks*, June/July 2004.

[12] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *Proc. of NSDI*, May 2005.

[13] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proc. of the 19th ACM SOSP*, October 2003.

[14] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlays. In *Proc. of IEEE INFOCOM*, March 2003.

[15] M. Castro, A. Rowstron, A.-M. Kermarrec, and P. Druschel. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication*, 20(8), October 2002.

[16] Y.-H. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early experience with an Internet broadcast system based on overlay multicast. In *Proc. of USENIX ATC*, June 2004.

[17] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proc. of ACM SIGMETRICS*, June 2000.

[18] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communication of the ACM*, 21(12):1040–1048, December 1978.

[19] S. E. Deering. Multicast routing in internetworks and extended LANs. In *Proc. of ACM SIGCOMM*, August 1988.

[20] C. Diot, B. N. Levine, B. Lyles, H. Kassan, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. In *IEEE Networks special issue on multicasting*, 2000.

[21] P. Druschel, E. Engineer, R. Gil, J. Hoye, Y. C. Hu, S. Iyer, A. Ladd, A. Mislove, A. Nandi, A. Post, C. Reis, A. Singh, and R. Zhang. Freepastry 1.3.2. freepastry.rice.edu, February 2004.

[22] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling and analysis of a peer-to-peer £le-sharing workload. In *Proc. of ACM SOSP*, December 2003.

[23] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable multicasting with an overlay network. In *Proc. of the 4th USENIX OSDI*, October 2000.

[24] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proc. of the 19th ACM SOSP*, October 2003.

[25] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-ef£cient management of DHT routing tables. In *Proc. of NSDI*, May 2005.

[26] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In *Proc. of IEEE INFOCOM*, March 2005.

[27] R. Mahajan, M. Castro, and A. Rowstron. Controlling the cost of reliability in peer-to-peer overlays. In *Proc. of IPTPS*, February 2003.

[28] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Universal topology generation from a user's perspective. Technical Report BUCS-TR-2001-003, Boston University, April 2001.

[29] T. S. E. Ng, Y. hua Chu, S. G. Rao, K. Sripanidkulchai, and H. Zhang. Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In *Proc. of IEEE INFOCOM*, April 2003.

[30] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient peer-to-peer streaming. In *Proc. of IEEE ICNP*, November 2003.

[31] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proc. of NGC*, November 2001.

[32] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *Proc. of IPTPS*, March 2002.

[33] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proc. of USENIX ATC*, December 2004.

[34] A. Rowstron and P. Drushel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware*, November 2001.

[35] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer £le sharing systems. In *Proc. of MMCN*, January 2002.

[36] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proc. of ACM SIGCOMM*, August/September 2004.

[37] I. Stoica, R. Morris, D. Krager, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. of ACM SIGCOMM*, August 2001.

[38] M. Theimer and M. B. Jones. Overlook: Scalable name service on an overlay network. In *Proc. of ICDCS*, July 2002.

[39] Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogeneity into an advantage in overlay routing. In *Proc. of IEEE INFOCOM*, March 2003.

[40] M. Yang and Z. Fei. A proactive approach to reconstructing overlay multicast trees. In *Proc. of IEEE INFOCOM*, March 2004.