Virtual Direction Multicast: An Efficient Overlay Tree Construction Algorithm

Suat Mercan and Murat Yuksel

Abstract: **In this paper, we propose virtual direction multicast (VDM) for video multicast applications on peer-to-peer overlay networks. It locates the end hosts relative to each other based on a virtualized orientation scheme using real-time measurements. It builds multicast tree by connecting the nodes, which are estimated to be in the same virtual direction. By using the concept of directionality, we target to use minimal resources in the underlying network while satisfying users' quality expectations. We compare VDM against host multicast tree protocol. We simulated the protocol in a network simulator and implemented in PlanetLab. Results both from simulation and PlanetLab implementation show that our proposed technique exhibits good performance in terms of defined metrics.**

Index Terms: **Overlay multicast, path stretch, peer-to-peer, peer-to-peer TV.**

I. INTRODUCTION

RECENTLY emerged Internet applications such as Internet protocol television (IPTV) [3]–[5] tele-conferencing and online education requires group communication, also known as multicast. The fact that the Internet bandwidth has become capable of carrying data-rich multimedia applications brought the expansion of Internet usage as well. The Internet protocol (IP) convergence is progressing and content providers are increasingly transporting multimedia content over the Internet. Multimedia streaming and live video distribution applications such as IPTV, peer-to-peer TV (P2PTV) [6], [7] are already constituting a significant portion of the Internet traffic and expected to grow further in near future. This unavoidable trend of converging video and multimedia traffic on to the Internet is calling for mechanisms with efficient and scalable transfer of content to many receivers from a single source. Such content delivery to many receivers is desired to be seamless to the multi-provider operation of the Internet.

Many researchers have put their research focus on achieving a robust and efficient way of sending traffic via multicast. Network layer multicast, a.k.a. IP multicast [8] attracted the attention for years; however, it has not become a widely used protocol because of its various deployment issues. IP multicast was proposed to provide efficient group communication, and can be implemented by integrating additional algorithms and tables to routers. Internet service providers (ISPs) are reluctant to support IP multicast because it introduces extra workload and complicates network management. Since IP multicast does not get much support from network operators, application layer multicast (ALM) [9]–[19] has emerged as a promising solution to achieve the multicast functionality. The idea is to establish a virtual network among end-hosts, each of which not only receives the stream but also forwards to other end-hosts. ALM does not require support from network layer routers. Only the end-hosts constitute multicast group, which moves functionality from the network layer to the application layer. This makes ALMs easy to deploy across multiple ISP domains and underlying network technologies. Although backbone-level multicast streaming applications such as IPTV will serve a particular need, overlay multicast streaming will likely be the ubiquitous solution to the multimedia delivery to the end points.

Among other things, the key to the efficient overlay multicasting is to constructing an efficient and robust overlay multicast tree, which is a challenging task. An inefficient overlay tree may cause multicast traffic to traverse the same underlying physical links multiple times and defeat the whole purpose of multicasting. Likewise, an unhealthy overlay tree may cause significant disconnections and traffic loss when failures happen in the overlay and underlying network. Being able to respond to application requirements and goals while keeping the overlay tree efficient from the network's perspective is one of the goals of this work.

In this paper, we propose a new P2P multicast streaming technique, called virtual direction multicast (VDM). VDM focuses on tree construction method to reduce redundant data transmission and failure recovery to decrease data reception outage under churn. We aim to find the most appropriate parent for a peer so that data travels the minimum possible path. To converge on a tree with a minimal source-receiver delay, we exploit round-trip times (RTTs) to measure "virtual distances" between peers. VDM uses an iterative approach by selecting a child which is in the same "virtual direction". The iterative process continues until the best potential parent is found. The key idea is to connect the nodes which are in the same virtual direction so that we try to minimize the source-destination path length for the overall multicast structure.

Key contributions and findings of our work are as follows:

- *Virtual directions as a multicast embedding.* We introduce the idea of using virtual directions as an embedding to establish overlay multicast trees. Even though the concept of virtual directions was used for routing P2P networks earlier [20] using the concept for establishing multicast trees was not tried before.
- *Multicast tree construction using virtual directionality on a*

Manuscript received August 26, 2014; approved for publication by Choong Seon Hong, Division III Editor, August 5, 2015.

Parts of this work was published as conference papers in [1] and [2].

This work was supported in part by US NSF award 1321069.

Suat Mercan is with the Computer Engineering Department, Zirve University, Kizilhisar, Gaziantep, 27260, Turkey. email: suat.mercan@zirve.edu.tr.

Murat Yuksel is with the Computer Science and Engineering Department, University of Nevada Reno, Reno, Nevada, 89557, USA. email: yuksem@cse.unr.edu.

Digital object identifier 10.1109/JCN.2016.000060

line. We detail an overlay multicast tree construction algorithm, VDM, using virtual directionality on one dimension, i.e., a line. We inspect each possible case and illustrate how an arbitrary overlay tree could be embedded as a set of one dimensional relationships of virtual directions. We compare VDM's performance against the most similar overlay multicast technique, Hessian message transport protocol (HMTP), that uses delay-based proximity of the nodes to establish the tree. Our results show that one dimensional embedding can successfully outperform HMTP by using only one dimensional virtual directions.

- *Virtual directions customized for application needs.* We design a way of generalizing the "distance" on virtual directions and abstract different metrics (delay or loss) to express the virtual distance. Via this generalization, we show that the overlay trees being calculated by VDM can be customized for application sensitive to different performance metrics such as delay or loss.
- *Distributed implementation of VDM.* We detail how join and leave procedures could be done for VDM. We show that VDM can survive well against churn in the P2P network. Further, we implement VDM on PlanetLab and experiment with real traffic streams showing its sustained performance on realistic settings like the PlanetLab.

We organize our paper as follows: We start with a comprehensive discussion of key issues in designing overlay multicast schemes and survey related work in Section II. Section III gives a detailed description of our VDM protocol. Then, VDM is compared to HMTP. Simulation setup and results of a comparative performance evaluation of VDM are presented in Section IV. Section VI presents implementation and results on PlanetLab. Finally, in Section VII, we summarize our work with conclusions.

II. RELATED WORK

A. Overlay Multicast Issues

ALM is flexible and easy to deploy, but its performance heavily depends on how the overlay multicast tree is constructed. The common goal of all ALM methods is to obtain an efficient and robust overlay multicast tree. However, the criteria for effectiveness of the overlay multicast tree can be various depending on the application goals. For example, live multimedia streaming is a real-time application that requires *minimal delay*, where the delay is defined as the time needed for a packet to reach its receiver(s). The data packets should ideally traverse the minimum path while being transferred from the source to the destinations; however, the end-to-end delay might be longer due to a high number intermediate nodes as a result of an inappropriate overlay structure. For such delay-sensitive applications, the overlay tree design should provide the minimum possible delay for each multicast receiver.

Another challenge to be addressed in an ALM system is the ad-hoc behavior of the members of the overlay tree. This is particularly a major issue for P2P scenarios where members of the overlay tree are not obligated to stay in the tree. Since most of the P2P systems do not have membership requirements, peers might join and leave at any time. This behavior, known as churn, makes tree maintenance harder. Ungraceful exit made by a peer may cause interruption of data reception at its descendants. When such ungraceful exits happen, the orphan peers need to be quickly reconnected to another parent. Long and frequent data outage is not acceptable for real-time applications, and thus *robustness against churn* is of crucial importance for overlay multicasting.

Moreover, large volume of data is transmitted in multimedia applications, which requires *avoiding redundant transmission* of the multicast traffic. The reason for client/server model not being feasible for these applications is that the data traffic has to be sent to each receiver separately which uses up bandwidth and server power. IP multicast is the best solution from this perspective, if we disregard its deployment shortcomings on the Internet. IP multicast prevents duplicate transmissions since it constructs the multicast tree at the router level. So, ALM cannot solve this problem as optimum as IP multicast, and causes redundant transmissions due to overlapping of overlay links on the same router level links. It is crucial to minimize the amount of such redundant transmissions by efficiently constructing and maintaining the overlay multicast tree.

One of the drawbacks of ALMs is being deprived of underlying network structure knowledge. This makes it hard to construct efficient multicast data paths. This could be solved by doing some on-the-fly measurements. Most commonly used technique is to measure distances between peers as RTTs. Some geolocation techniques which estimate geographical location of an IP address, topology maps and network coordinate systems also can be used to overcome this problem. In general, measurement of underlying network and utilization of this information to construct overlay tree is a key issue in ALM design.

B. Previous Work

Since the emergence of ALM concept, numerous algorithms have been proposed using different techniques to achieve a successful overlay tree for live video streaming. Overlay network construction techniques can be classified into two main categories according to their structure [21]: mesh-based and tree-based.

In the mesh-based approach, either nodes join to multiple disjoint trees (e.g., SplitStream [15] and ChunkySpread [16]) or choose a set of neighbors to create a mesh topology (e.g., CoolStreaming [22], Narada [10], MeshTree [23]). This approach is known as a pull-based mechanism. An important characteristic of the mesh-based approaches is their robustness to churn; but they are more costly to maintain due to the higher control overhead, which also limits scalability. They trade off more robustness with more overhead. In general, the mesh-based approaches are not satisfactory in terms of network resource usage and are wasteful in leveraging the underlying network bandwidth.

In the tree-based approach (e.g., BTP [13], HMTP [14], Yoid [9], TAG [24], OMNI [25]), nodes are organized in a tree structure rooted at the source. The nodes have parent-child relationships. When a node receives a packet from its parent, it forwards to children; which is also considered as a push-based mechanism. The tree is extended when a new node joins the group. The tree-based approach is efficient in terms of avoiding redundant

data transmissions; but, when a node leaves its offsprings and peers may suffer from data outage. The tree must be repaired quickly to reduce the data loss. So, the tree-based approaches are not robust to churn relative to the mesh-based approaches. Another disadvantage is that while interior nodes are busy with forwarding data, leaf nodes stay idle; which is an unfair manner for members. The biggest advantage of the tree-based approaches is their small maintenance overhead, which allows them to scale to very large groups.

The tree-based approaches can be further subcategorized in three different dimensions: single vs. multiple trees; single or multiple layers in the data traffic; and with or without super nodes. Multiple tree approaches establish multiple overlay trees to deliver the data traffic. The approaches with multiple layers code the multimedia data traffic into layers where the layer 0 provides the lowest quality and higher layers add more quality/resolution to the received traffic at the destinations. The approaches with super nodes assume that some nodes in the overlay network are willing to undertake more responsibility and are more robust; and thus tailor the overlay tree construction mechanism based on the existence of such nodes.

Our approach, VDM, is a *single tree, one layer* algorithm *without super nodes*. We focus on the most fundamental problem of establishing a single tree with minimum number of redundant (overlapping) links and minimal delay from the source to the receivers. The closest prior work to ours is HMTP [14], which aims to solve the same problem. We give a short description of HMTP here. We will discuss the differences in detail later in subsection III-F. HMTP interconnects IP-enabled islands. If IP multicast is available in any subnet, one node is selected as head to join the overlay tree and IP multicast is used in subnet. The key idea in HMTP is connecting nearby peers. When a new peer wants to join, it contacts the source, and get list of the children. By probing each child, it finds closest child to itself in terms of delay. It repeats the same process with the closest child. This iterative process is repeated until best potential parent is found. HMTP also applies a tree refinement process. Each node randomly selects a peer in its root path and look for if any closer peer than its parent connected in meantime. This refinement process is repeated periodically. HMTP aims to reduce routing inefficiency. It also proposes a foster child concept to shorten startup time. A node connects root at the beginning to start stream immediately. Then, it jumps to ideal parent when it is found.

There also have been many proposals for establishing virtual coordinate systems, e.g., [32]–[35]. They are widely used in P2P systems either for file sharing or multicasting. These coordinate systems rely heavily on the accuracy of reference points. Basically there exist landmark-based and decentralized approaches to build the map. In the first one, an infrastructure node is used as the reference point while any node can be used as the reference in the second approach. A new node contacts these reference points and tries to locate itself. Estimating location of peers will help the system to optimize the performance, but it will also require 2D or 3D embeddings of the network. So, virtual coordinate systems are a stronger form of representation than our "virtual directionality" embeddings of the network. Although they will certainly result in better and more accurate representation of the multicast tree at hand, they will also require more messaging among siblings to figure out the 2D or 3D coordinates. Our algorithm is not a coordinate system. But it tries to locate peers by utilizing directionality concept with an iterative method. Our goal is to reduce the complexity of 2D or 3D virtualization of the network to a 1D space. Our key contribution is to show that 1D space can be effective, but may not be optimal, in virtually embedding a network.

III. VIRTUAL DIRECTION MULTICAST

Overlay multicast is an application layer technique that establishes a virtual network by connecting end hosts using logical links. Different dynamics play key role in designing an overlay tree in such a virtual network. Our goal is to build a virtual network confined to physical network. Even though our ultimate goal is not to find minimum spanning tree (MST) for overlay tree, we try to converge to MST as much as possible by using local and simplistic methods that can be practical to implement.

One reason to use overlay multicast instead of unicast is to relieve the network from redundant traffic. So, establishing a multicast tree close to MST is important, but not the only and ultimate goal. The overhead messages for constructing the tree should not overwhelm the system, which would destroy the real purpose of the overlay. On the other hand, the system design should take other performance factors which are important for users into account.

VDM is an overlay multicast algorithm. It builds a multicast tree by making parent-child relationships between nodes which are determined to be on the same virtual direction based on performance of the connections between them. VDM uses a single tree for multicast purpose. Each node has only one parent, but might have more than one child.

It aims using resources more efficiently by building the multicast tree in a reasonable way by measuring inter-peer distances in a one dimensional directional abstraction and by using fewer number of maintenance messages. It uses a decentralized method for tree construction. Each peer contacts the source at the beginning and finds a proper node to connect. VDM also observes user expectations by trying to reduce startup time and reconnection time.

A. Protocol Description

A.1 Key Design Considerations

In an overlay network, converging to MST while observing other multicast requirements should result in better performance in terms of resource utilization and overall multicast quality. Overlay network is a degree-constrained environment. Each node has a certain number of outgoing links, and thus, the multicast tree must be constructed within this degree constraint. Degree-constrained minimum spanning tree (DCMST) problem is known to be NP-hard [26]. Additionally, in a peer-to-peer network environment, the overlay tree changes because of constantly new coming and leaving nodes. Moreover, network dynamics causes changes in path performances between nodes and may require reconstruction of the overlay tree for better performance.

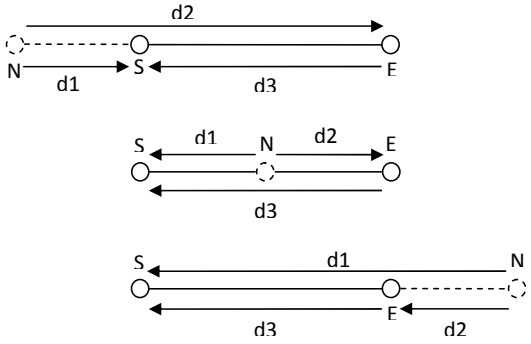

Fig. 1. Directionality on a line.

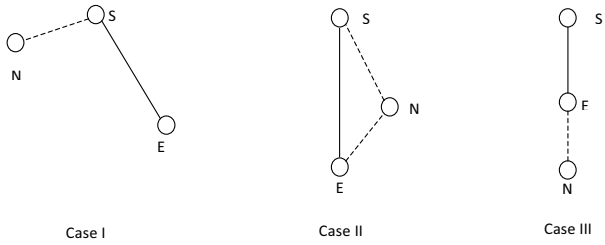

Fig. 2. Directionality concept on 2D.

When we think all of these, calculating global MST is expensive and difficult. But, staying close to MST using simple methods while satisfying other requirements is a better choice. With using VDM that calculates virtual distances between the overlay nodes in a 1-D space, we try to converge to MST in general. In order to achieve this "directional" abstraction, we define three succinct cases that will be explained later.

A key design component of VDM is *directionality*. We locate a newly joining peer relative to existing peers with an iterative process using this concept of directionality. We take the peers three by three, and we estimate the location of the new peer relative to the existing peers by comparing inter-peer distances.

In environments like P2P networks, churn is a major issue. When peers are leaving or joining frequently, the performance of the protocol depends heavily on being able to swiftly switch to a new tree. Reevaluation of the overall multicast tree requires a centralized approach and is typically not possible within the very short period of time available for switchover. Our directionality-based procedure is completely distributed and can quickly establish a new and good performing tree with local repair.

A.2 Virtual Directionality on a Line

VDM exploits virtual directions in order to organize nodes. Suppose that there is a source, S, and an existing, E, node which are already in overlay network. A new node, N, is going to join the overlay tree. We measure the distances among these three nodes. N could be in three different positions according to S and E. Three nodes can form three combinations in linear representation, i.e., a line.

Distances d1, d2, and d3 are RTTs measured by probing. Longer distance is generally not equal to the sum of shorter distances which seems equal in linear representation. We look at the longest one to determine into what case the combination falls. There are three cases:

Case I: *Source is in the middle.* In this case, a new direction is created for N.

Case II: *The new node is in the middle.* N is placed between S and E.

Case III: *The existing node is in the middle.* N is in the virtual direction with S and E.

In order to make virtual direction concept more understandable, we try to show it on 2D in in Fig. 2. Dashed lines are the ones that will be added. With this technique, we aim to minimize multiple packet transmission on the same link and resource usage in the network. If there has to be multiple packets on a link, we try to find possible shortest one to minimize network usage.

B. Join Process

Algorithm 1 Join Procedure

```
1:  S ← source
2:  N ← newnode
3:  Contact(S)
4:  N pings S and all children of S
5:  D(n) ← Directional nodes
6:  if D(n) is not empty  (Case II or Case III exist) then
7:    if D(1..n) is between S and N  (All in Case III) then
8:      S ← closest(D(n))  (Select closest of Case III)
9:      Contact(S)  (Continue from closest one)
10:   end if
11:   if N is between S and D(1...n)  (All in Case II) then
12:     for D(1..n) do
13:       if N has free degree then
14:         S becomes parent of N
15:         N becomes parent of D(i)
16:         Update grandparent of D(i)'s children
17:       end if
18:     end for
19:   end if
20:   if Case II and Case III together then
21:     S ← closest(D(n))  (Select closest of Case III)
22:     Contact(S)  (Continue from closest one)
23:   end if
24: else
25:   if S has free degree then
26:     N connects to S
27:   else
28:     N connects to closest free child
29:   end if
30: end if
```

A pseudo-code for the Join procedure is given in Algorithm 1. Nodes store some state information to cope with the protocol. Each node has children list and distances to them. They also know their parent and grandparent. When a node wants to join to the overlay tree, it sends a connection query to the source. It gets the list of its children and learns RTT by probing to each child. During a join process, we first look for if

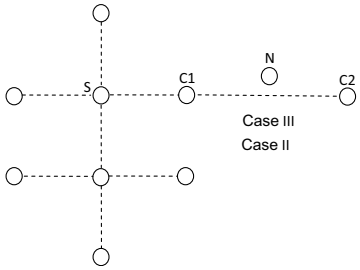

Fig. 3. A join example.

Case II or Case III exists among parent, an existing child and new child. We may find only one of these cases, multiple of the same case or two of them together.

If Case II or Case III is not found, it means that the new node is not in the same direction with any of existing children in this iteration. Then, it connects currently queried node if it has a free degree (or outgoing interface). Otherwise, it connects to the closest free child.

If we encounter Case III, we proceed to next iteration from that child, and repeat the same procedure. If we have Case III with more than one child at the same time, we select the closest one.

If we find Case II which means the new node is between two existing nodes (parent and currently checked child), then proper connections are made, and join process is done. In some cases we might have Case II with more than one existing child. Then, we make connections as long as the new node allows. Since every node stores grandparent information to use in case of parent failure, grandparent information of existing child's children should be updated.

If we find Case II and Case III together, we continue with Case III by selecting the closest one if more than one Case III is existing.

When N eventually finds the correct node, it connects and joins the session. A node can accept connections up to its maximum degree, which we call "degree_limit". Each node has a pre-defined degree_limit. We assume that degree_limit of each node is at least one. If the potential parent that new node decided to connect reached its degree_limit, new node connects to its closest child which can accept connection without breaking its degree_limit. The degree_limit expresses how much the node is willing to take other peers' load. Free-riders, for example, have low (typically 1) degree_limit.

B.1 Join Example

We now illustrate how the cases defined above are utilized for the join process in VDM. Let's assume that we have an existing tree with several children as in Fig. 3. S denotes the source and others are children in the tree.

N in Fig. 3 attempts to join the multicast session. N contacts the source by sending *information_request* message. Source replies this message with *information_response* which contains children list and distances to them. N queries all children to get distance information. When N starts receiving *information_response* messages from children, it starts to check described cases. N does not have any information about C2 at the beginning. It detects C1 which falls into Case III. After that N continues its join process through C1. It gets children list from C1. It receives distance information from C1's children. While checking these children, it will see that N is between C1 and C2 which we call Case II. So, N will connect to C1 while C2 changes its parent from C1 to N. When the proper connections are made, connection process is done.

C. Complexity Analysis

In this part, we analyze the complexity of our join algorithm. A node who wants to join the session first contacts the source and gets all children information. Based on this information, it determines a direction to go. It will pick up one child and repeat same procedure through that child if needed. This process is repeated until the best potential parent is found.

For our analysis, we assume that each child has the same number of degree and it is a balanced tree. Let's say n is node degree for one peer, N is the total number of nodes and a is tree depth. It is possible to express the relationship between the number of nodes and the rest of the parameters as:

$$N = n^a. \tag{1}$$

Then, the tree depth will be

$$a = \log_n(N) \tag{2}$$

which is in the order of $O(\log N)$.

In the worst case, if the node will join the tree at the leaf, the number of nodes it has to contact, A, will be

$$A = n\log N. \tag{3}$$

Since the time duration of the join procedure is directly dependent on the number of nodes being contacted by the new node, the complexity for join algorithm will be in the order of $O(n\ \log N)$ which provides scalability and short startup time for nodes. Note that this worst case complexity is valid for scenarios with the maximum node degree being n as well. For such cases, the average time complexity will be lower than $O(n\ \log N)$.

D. Reconnection

An overlay network is an ad-hoc environment, particularly when it is a peer-to-peer overlay. The system depends on users who receive the stream through each other. The users are free to join and leave the system at anytime. Even though some incentive and punishment mechanism can be utilized to increase the stability, adhocness (or churn) is still in the nature of the system.

Our algorithm builds a tree to transmit the streaming traffic flow from source to each user. In this context, when a node leaves the session, some orphan nodes occur within the tree. These nodes have to find a new parent to continue receiving data.

In VDM, a peer is required to inform its children when it is leaving. When an orphan child gets this leave message, it starts the join process at its grandparent, Fig. 4. Since our algorithm is using a single tree, quick reconnection is important to avoid high loss. We start reconnection process at the grandparent instead of the source to expedite the reconnection process. In that sense,

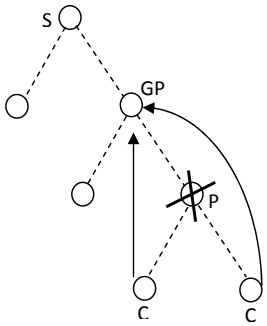

Fig. 4. Orphan nodes starts reconnection at grandparent.

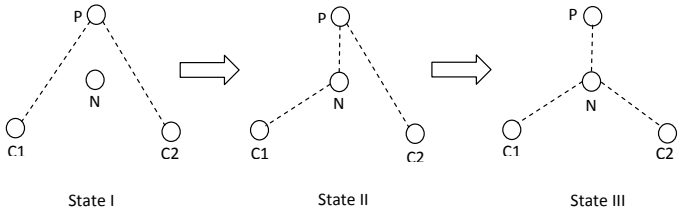

Fig. 5. Case II is existing with two different children in the same iteration.

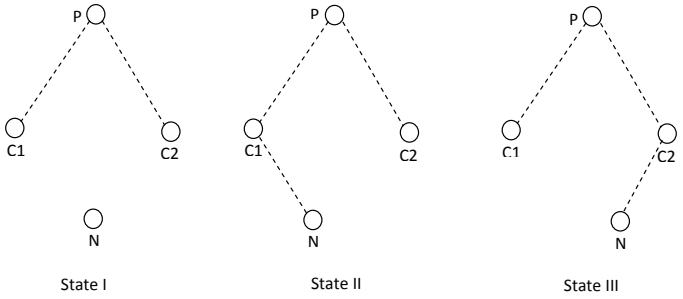

Fig. 6. Case III is existing with two different children in the same iteration.

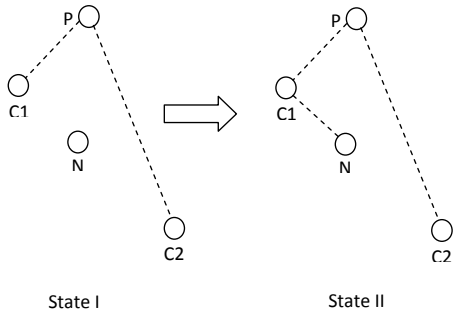

Fig. 7. Case II is existing with one child and Case III is with another child in the same iteration.

reconnection is basically same thing as the join process except that it starts at the current grandparent. If both the parent and the grandparent leave at the same time, which could occur very occasionally, the orphan node goes to the source for reconnection. Since the reconnection starts at the grandparent, we expect that it is accomplished in a very short period of time compared to regular join process.

E. Corner Cases

In subsection III-B, we presented a simple example which includes only one case at a time to explain VDM's join process. We now explore exceptional scenarios to understand some corner cases take place where two cases (Case II and Case III) can exist at the same time.

E.1 Scenario I: Two Case IIs

In this situation, N detects Case II with two children, Fig. 5. N selects C1 or C2 as a child and P becomes parent. Then, if N has a free degree, the other child connects to N. So, we get the best solution in terms of local MST. Normally, in order to get this solution, we should know the distance among all nodes. In this case, we know all distances except C1-C2.

In order to know the distance between C1 and C2, (i) either we should store this information (distance to sibling) and update it when necessary or (ii) we should make another measurement among siblings of P at the time of join. Nodes in the overlay join at different times. We measure the distance, C1-C2, when C1 or C2 is joining the tree. Then using this information we make the connections. But, after that we don't save the distance information. The former approach, storing this information, increases the amount of state information to be stored. Further, this information has to be updated all the time with new coming or leaving siblings. All the siblings of a node might change with a parent change which occurs if a node get connected with Case II in between child and parent. This complicates maintaining this state.

Second choice, new measurement among siblings, increases the overhead of the system. For each join process, siblings should communicate with each other.

In our proposed system, using 1-D abstraction, we reach desired tree without storing extra information or using redundant messaging. This case is handled in lines 9–11 in Algorithm 1.

E.2 Scenario II: Two Case IIIs

In this situation, Fig. 6, N detects Case III with two children. We select the one which is closer to N. Then, we continue the join process through that node. This case is handled in lines 7–9 in Algorithm 1.

E.3 Scenario III: One Case II and One Case III

In this situation, N has Case III with C1 and Case II with C2, as shown in Fig. 7. In such a conflict, we choose Case III and continue join process from C1.

This situation is a scenario that misses local MST in Fig. 7. This problem can be resolved by using some more specification in the design. When we do this, C3 in Fig. 8, if there is such a node, is not able to find C2. So, it might cause other problems. We don't want the directions to diverge too much from its original definition. As a result, we intentionally leave Scenario III as it is. This case is handled in lines 20–22 in Algorithm 1.

E.4 Scenario IV: Hidden Grandchildren

Another example such a scenario where the joining node may not find the closest node is presented here. In Fig. 9, the best potential parent for N is C2. But, when N checks P's children, it will see C3 as a directional node and miss C2. This situation can be prevented only by contacting grandchildren of P which increases the overhead and time.

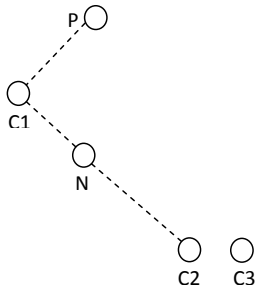

Fig. 8. C3 is not able to find C2 directionality divergence.

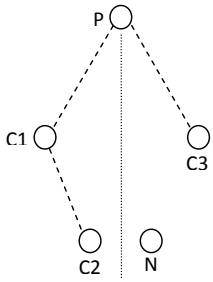

Fig. 9. N needs to contact grandchildren of P to find C2.

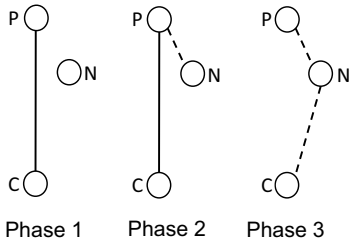

Fig. 10. A scenario showing the difference between VDM and HMTP.

F. VDM versus HMTP

Overlay multicast is a well-studied area. Researchers have developed numerous numbers of algorithms that have different approaches as we mentioned some of them in Section II. All of these techniques have their own advantages and disadvantages. HMTP is one of these proposed techniques. Even though we didn't get inspired from this technique, we found that it is similar to our proposed algorithm VDM. So, it is important to make one-to-one comparison between VDM and HMTP to see the differences and our superiorities.

VDM and HMTP use different approaches when building multicast tree. HMTP focuses on closeness while VDM utilizes directionality concept. In HMTP, a node finds closest node to attach, then with periodic tree refinement process the tree is tuned. In our protocol, we try to detect nodes which appears in the same direction.

We look at Fig. 10. In phase 1, N comes to join to the overlay. With HMTP, N connects to P first, then C finds N by sending a refinement message to its parent to see if there is a closer node. The overlay tree is formed as in phase 3. But, by using VDM we can directly detect the case and make proper connections. The disadvantage of using HMTP here is that since the tree is degree constrained, desired connections may not be established. If P cannot accept N because of degree limitation, the opportunity to connect these three nodes in the best manner will be missed. Another disadvantage of HMTP is that it has to use periodic tree refinement to be able to detect closer child. Using our directionality concept, we directly connect these three nodes in best way without using any extra messaging.

VDM and HMTP have different refinement processes. HMTP selects one node on its path to source, and starts the refinement process from that node. HMTP uses refinement to complete the join process. A node in HMTP has to do refinement to find a closer sibling. But, VDM achieves the same thing without any refinement. This requirement for HMTP exposes too high overhead for the system. Our refinement purposes to adapt the tree to changing conditions of the internet.

Refinement is a part of the join process for HMTP. Every node has to check if there is a closer node. The time to converge to a better tree depends on the frequency of these refinement messages. On the other hand, VDM achieves better tree without using refinement messages. So, VDM is very efficient in terms of overhead when compared to HMTP.

IV. PERFORMANCE EVALUATION

A. Simulation Setup

We use NS-2 [27] to conduct simulation experiments for evaluating our protocol. We generated transit-stub model topology consisting of 2,576 nodes using GT-ITM [28]. One of the nodes is chosen as source for the multicast tree. The source is assumed to be alive during the entire simulation time, and is known by other peers. Randomly selected 500 of 2,576 nodes join to the overlay multicast tree. We run the simulation for 10,000 s, and dedicate 2,000 s for the join process at the beginning. We take 400 s as a time interval and define the churn based on that interval. Based on the churn rate, a number of nodes join and leave the tree. For example, if the churn rate is 5%, then 25 new nodes join and 25 of the existing nodes leave in each time interval. We uniformly pick the nodes to join and leave for modeling churn.

Even though previous work showed that most of the peers tend to continue staying in the overlay and that a small portion of the peers tend to frequently leave [29], we use a uniform probability of a peer node joining or leaving the overlay tree. The literature also showed that peers with more popular content tend to be more stable [30], which implies the multicast nodes closer to the source should be staying longer in the overlay. This further implies that the peers closer to the tree leaves are more probable to leave the overlay. So, our assumption of uniform sampling of peers for join/leave is a conservative one [31], since it allows testing of our protocols under adverse conditions where nodes in the middle (or closer to the source) of the tree can leave at an equal chance with the leaf nodes.

The number of nodes in the overlay is retained at 500 by the end of the 400 s time interval. At the end of every time interval, we give 100 s for tree to get stabilized, then we do the measurements. We expose the tree to churn again in the next time interval after the measurement. This process is repeated until the end of the entire simulation time. For instance, the nodes are renewed

almost twice over lifetime under 10% churn. Some nodes may join and leave several times while some never join. There is no super node in that all nodes are considered equal. Degree_limits of nodes are uniformly distributed within the range from 2 to 5.

We simulated the protocols under different churn rates from 1% to 10%. We repeated the simulation experiments 32 times for each churn rate, and we report 90% confidence intervals on our results.

B. Performance Metrics

We are interested in efficiency of data delivery path and service quality that end-users are experiencing. In order to quantify these two targets, we focus on four performance metrics. Stress, stretch and overhead are the major factors for data delivery efficiency. Service quality is basically measured with loss rate and delay.

- *Stress:* Stress is defined as the number of identical packets transmitted on the same link. In IP multicast, stress is always one since a packet goes through a link only once.
- *Stretch:* Stretch is the ratio of path length a packet is traveling in the overlay multicast tree to that of in unicast. Unicast is assumed to have optimal stretch.
- *Messaging Overhead:* We define overhead as the ratio between maintenance messages and data messages.
- *Loss Rate:* Loss rate at a peer is the ratio of number of lost packets to the number of packets supposed to be received in the peer's lifetime.

C. Simulation Results

We show results of previously defined four metrics with 90% confidence interval. VDM is compared to IP multicast with stress. It shows how much VDM converges to IP Multicast. Stretch is comparing VDM to unicast. Unicast provides smallest delay for peers. Overhead and loss cannot be avoided especially under adhoc behaviors of peers. But they should be kept minimal, and they should not increase exponentially with churn rate.

In Fig. 11(a), we show stress vs churn rate. Stress is one of the most important metric for resource usage efficiency. Average stress is around 1.6 for both VDM and HMTP. VDM gives slightly better results. Stress doesn't change significantly while churn rate increasing for both protocols.

Fig. 11(b) shows stretch vs churn rate. Stretch is important for efficient content delivery and efficient resource usage. VDM outperforms HMTP in terms of stretch. Average stretch is around 7 for VDM while it is around 12 for HMTP. Stretch is slightly increasing with churn rate for both protocols.

Fig. 11(c) shows average loss rate for all nodes. End users are especially interested in continuity and quality of streaming. High loss rate dissatisfies end users. Average loss rate is below 2% for VDM under 10% churn. In this simulation, we do not apply link error which causes packets to be lost. So, all packet loss is caused by disconnection of churn. That is why it is so small when churn rate is low.

Fig. 11(d) shows the comparison between VDM and HMTP for overhead. Overhead should be kept small to put less load to network. It cannot be prevented from increasing with an increase in churn rate, but it shouldn't be exponential. Fig. 11(d) depicts that overhead increases linearly as churn rate increases. Overhead is around 2.2% for VDM when churn rate is 10%. HMTP has to use refinement messages by definition which causes high overhead ratio.

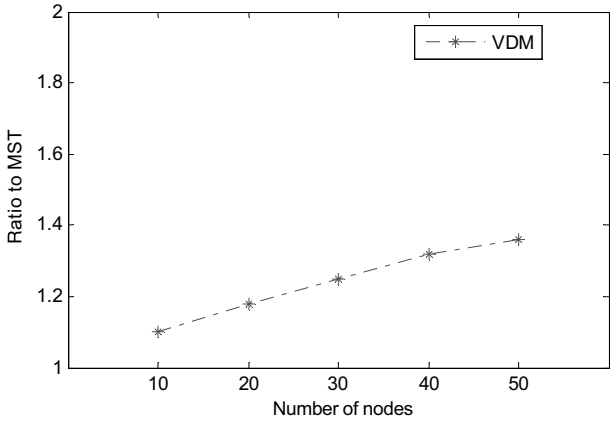

Fig. 12. Comparison with MST.

D. Comparison with Minimum Spanning Tree

With our algorithm we try to converge to MST while also trying to satisfy other user requirements. In this part, we test our algorithm to see how much it gets closer to MST. We do not expect VDM to find MST since there are some cases it does not guarantee reaching MST for simplicity. To observe the potential of VDM, we do not apply the degree limitation. Further, the degree-constrained MST is not practical [26]. Fig. 12 shows the ratio between the tree constructed by VDM and MST in terms of the total weight of the trees, i.e., the sum of link costs. As expected, the ratio is increasing when the number of nodes increases. The promising insight is that VDM stays close to MST with a ratio well below 2, and further, the increase in the ratio is sub-linear as the number of nodes increases.

V. GENERALIZATION OF VIRTUAL DISTANCE

Live multimedia streaming is a real-time application that requires *minimal delay*, where the delay is defined as time needed for a packet to reach its receiver(s). The data packets should ideally traverse the minimum path while being transferred from source to destination. Overlay tree design should provide possible minimum delay for each multicast receiver. However, delay and loss rate between two nodes may be uncorrelated because of background and cross traffic on routers. So, a peer might experience high loss rate on a good path in terms of delay. Sensitivity of multimedia applications differs against various network performance metrics such as delay, loss, or bandwidth. This requires taking other factors into account when building overlay tree.

A key property of VDM is the capability of virtualizing the underlying network in different ways. It is possible to establish "virtual directions" based on performance metrics delay, loss or bandwidth. Different values of these metrics may produce different virtual distances and thus different overlay tree in our protocol. By generalizing and customizing virtual direction, we can

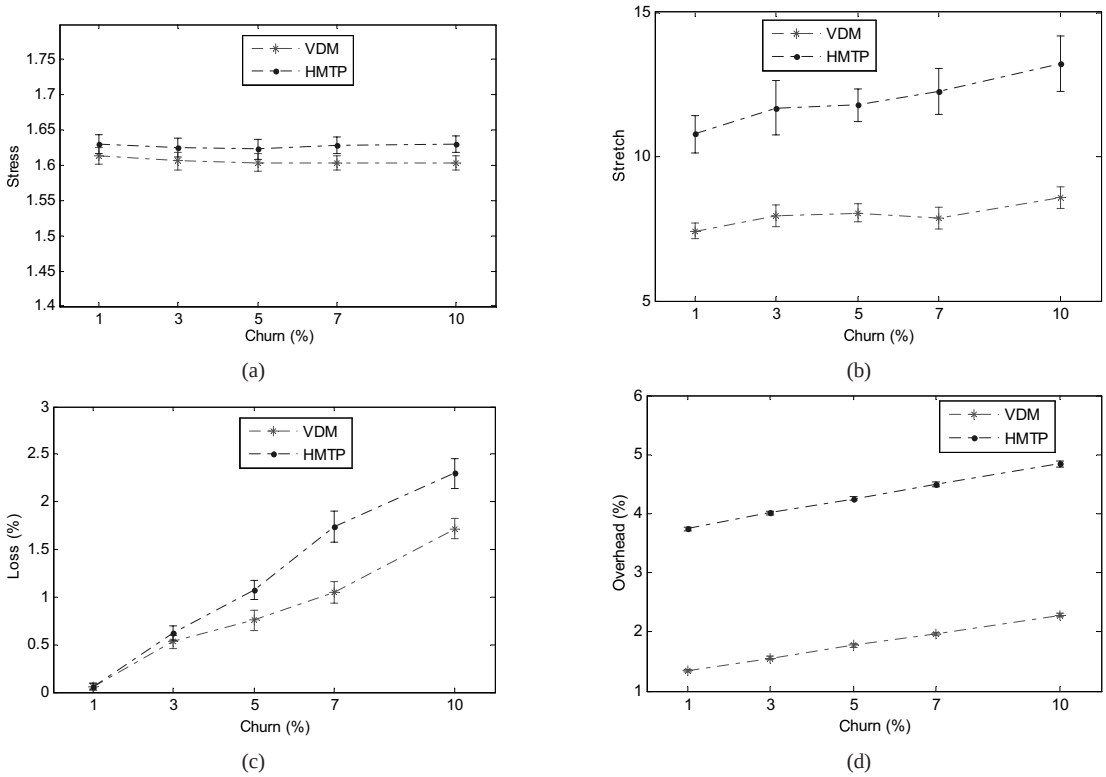

Fig. 11. VDM vs. HMTP: (a) Stress vs. time, (b) stretch vs. churn, (c) loss rate vs. time, and (d) overhead vs. churn.

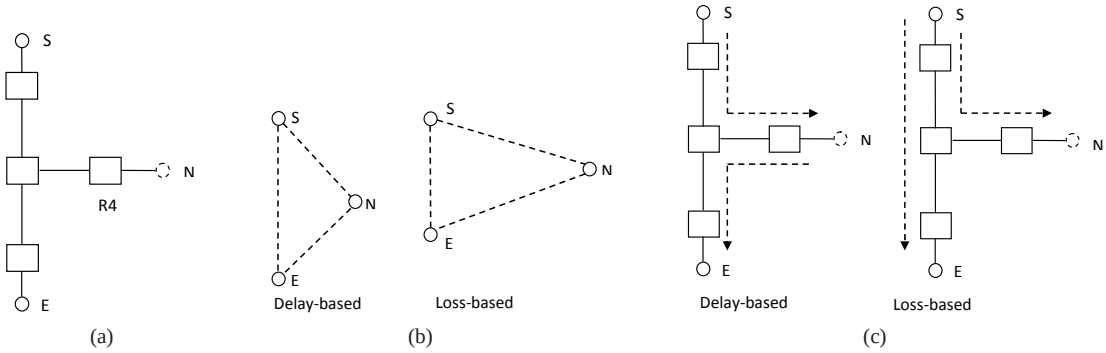

Fig. 15. Overlay topology construction based on delay and loss: (a) A sample topology, (b) relative virtual distances, and (c) differently formed overlay trees.

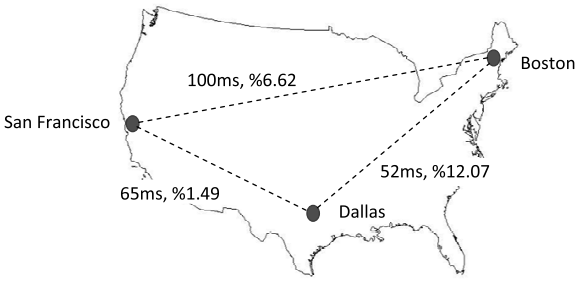

Fig. 13. Delay and loss rate measurement among San Francisco, Boston and Dallas

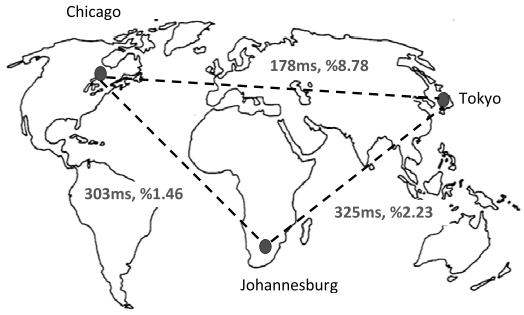

Fig. 14. Delay and loss rate measurement among Chicago, Tokyo and Johannesburg

establish target specific overlay trees to improve some specific performance metrics desirable by applications. Calculating the virtual distance based on different criteria, but without protocol modification, makes the overlay multicast protocol satisfy different quality expectations. Our key goal in this part is to automatically calculate overlay multicast trees such that they can be seamlessly customized to applications' performance goals.

In order to corroborate the generalization method, we took

simple measurement statistics from [36]. It shows latency and loss rate among three cities in United States and in three different countries. Values among San Francisco, Boston and Dallas are shown in Fig. 13. Ratio among three values for latency and loss rate is different, thus overlay tree to be constructed among three nodes in these cities will be different. As another example, we look at the measurements among Chicago, Tokyo and Johannesburg, values are shown in Fig. 14, which also gives different ratio for latency and loss rate.

We also took sample inter-PoP measurement dataset from [37] which has latency and loss rate information. From this dataset, we pick three points A, B, and C among the links whose loss rate is not zero. We look at delay of A-B (d1) and B-C (d2), and loss rate of A-B (l1) and B-C (l2). When we compare the ratios d1/d2 and l1/l2, 44% of this dataset is inversely correlated. And, the rest does not give the same ratio.

We illustrate a topology in Fig. 15(a). S is source, E is existing child and N is a newcomer. Relative distances among these three nodes might be different as shown in Fig. 15(b) when we do distance measurement in terms of delay and loss. As a result, overlay tree will be formed in different ways as in Fig. 15(c). For this specific topology, this difference is caused by the traffic characteristic on router R4.

We propose a generalized method of calculating overlay trees to increase user-perceived quality of performance-sensitive applications. We define and use the concept of "virtual distance" to determine "virtual direction" for constructing overlay trees. Abstracting applications' sensitivity within the virtual distances, we aim to find the most appropriate parent for a peer according to the application's purpose. We embed the virtual distance method in our protocol, VDM, and show that the protocol automatically calculates overlay trees based on delay (VDM-D) or loss (VDM-L), depending on which is more important for the application under consideration.

A. Performance Evaluation of the Virtual Distance Concept

We evaluate the performance of VDM-D (delay-based) and VDM-L (loss-based) in order to show the efficiency of the virtual distance concept in automatically customizing the overlay tree for application-specific performance goals. We analyze protocol behaviors as we vary the churn rate in the overlay network.

We use a similar setup with previous one for simulation except that each physical link in topology is assigned a random error rate between 0% and 2%. In this case, the topology consists of 792 nodes, and randomly selected 200 of the 792 nodes constitute the overlay multicast tree, and repeated the simulation experiments 10 times instead of 32. We decreased network size and number of experiments because we have loss in this case and we had to send much more packets from source which increased the size of trace file dramatically.

B. Simulation Results

We present results for performance comparison between overlay trees constructed using VDM-D and VDM-L. We investigate the behavior of these metrics versus churn rate for both protocols. The variations especially in stretch and loss rate based on utilized technique will affirm our proposal. We expect that VDM-L reduces loss rate while trading off stretch and that VDM-D gives better results for stretch.

In Fig. 16(a), we show stress vs churn. Stress is one of the most important metrics for resource usage efficiency. Stress does not change significantly while churn rate increasing. Average stress is around 1.5 and 1.7 for VDM-D and VDM-L, respectively. The closeness of the stress for the two protocols is expected as the virtual distance does not focus on the stress. However, it is also expected that stress is a little higher in VDM-L since delay is known to be more correlated with the physical distance between nodes, and thus the overlay tree becomes closer to the IP multicast tree in VDM-D.

In Fig. 16(b), we show stretch vs. churn. Stretch is important for efficient content delivery and efficient resource usage. Stretch does not get affected much with churn rate. Average stretch value is around 4 for VDM-D while it is around 7 VDM-L. We can infer from the graph that path length to source for end-users is reduced when using delay as basis for distance calculation. The results in Fig. 7 shows a clear differentiation of the overlay tree based on which metric, delay or loss, the user application might choose.

Fig. 16(c) shows average loss rate vs. churn. End-users are especially interested in continuity and quality of streaming. High loss rate dissatisfies end-users. Loss in this graph is caused by packet drops over path and disconnection because of churn. Packets are dropped over links according to their error rate. Churn causes the loss rate to increase. The graph proves that VDM-L improves the loss rate compared to VDM-D. Loss rate for both could be considered high, but each link in this setup is assigned a loss rate on purpose as we wanted to observe how much customization the virtual distance concept can achieve in terms of loss rate. To give an idea about how much excessive loss the results have, we have plotted another result for "VDM" where link error rates are set to 0%. In this case, loss rate is caused only by disconnections and is relatively low compared to the other two cases.

We finally looked at the overhead. We used the same number of probing messages to measure delay and loss rate even though delay can be measured with less number of messages. The overhead for VDM-L is a little less the the VDM-D, because the number of lost packets in VDM-L is fewer which makes denominator greater in the definition. Fig. 16(c) depicts that the overhead increases sublinearly as the churn rate increases since the nodes send additional probing messages to be able to reconnect.

Our concern was to obtain better performance results for certain types of metrics which may be more important for different applications. When we think of all the results together, VDM-D, uses delay for distance estimation, and improves stress and stretch while giving higher loss rate. It could be used for delay sensitive applications. On the other hand, VDM-L achieves better performance in terms of loss rate. It can be chosen for more delay tolerant but loss sensitive applications such as peer-to-peer file sharing.

VI. PLANETLAB IMPLEMENTATION

We implemented VDM on PlanetLab [38] to be able to test it under more realistic environment. For our experiments, we only

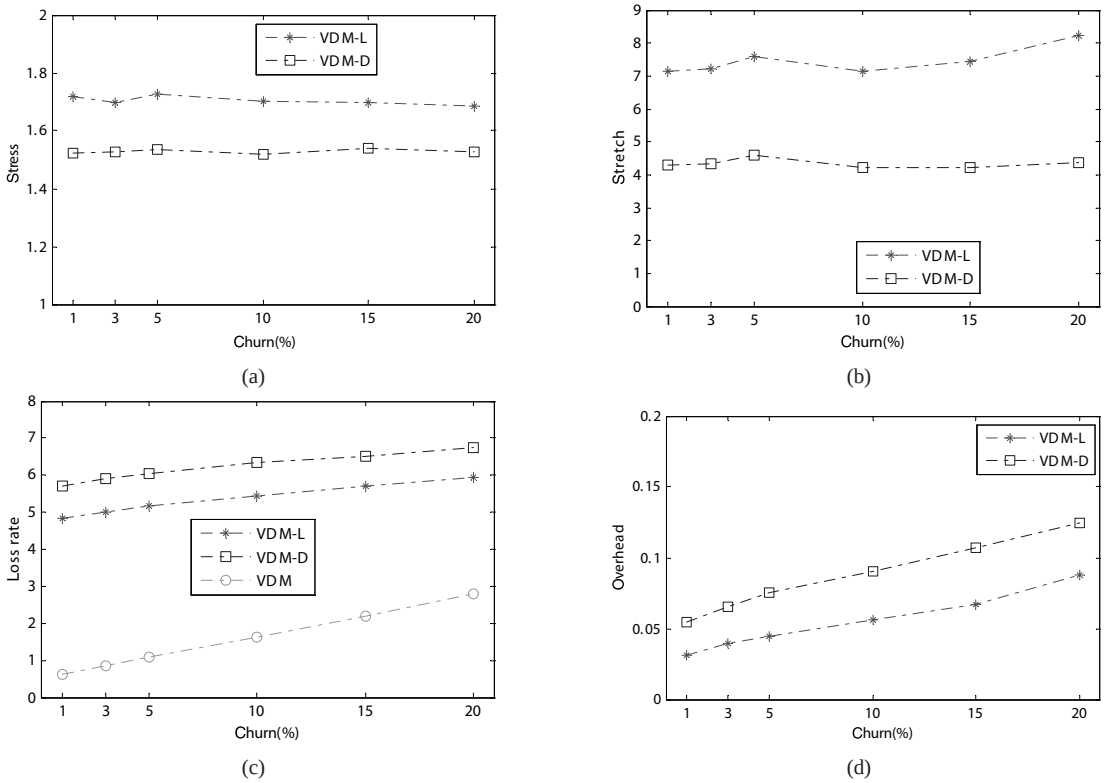

Fig. 16. VDM-D vs. VDM-L: : (a) Stress vs. churn, (b) stretch vs. churn, (c) loss rate vs. churn, and (d) overhead vs. churn.

used nodes in the United States. Also, we only use one node at one site. To be able to get accurate results we applied several filtering processes to select working nodes.

Main components of the implementation are illustrated in Fig. 17.

Scenario: There is a list of nodes that works properly on our local machine. We have a scenario generator which generates the scenario file by using the node list and provided seeds. A line in scenario file mainly has action type (join, leave), node information and time for action.

VDMAgent: VDMAgent is uploaded to every node on Planetlab. It carries out the core job of the protocol. When it receives the connect message from the main controller, it contacts the source which is known by each node. Then, VDMAgent using the algorithm embedded inside finds an appropriate parent to connect. After a node gets connected, VDMAgent keeps running and responds incoming information messages or connection requests.

Sender: Our system has single source which hosts the main data stream. The source node sends this data to its children. Other nodes transmits the data that they receive from their parents. Sender is responsible to send the data at the origin.

Transceiver: Every node has this unit except the source. When a node connects, the transceiver unit is started. The main job of this unit is to catch incoming data messages and transmit them to its children.

Main Controller: The Main controller is in charge of applying the scenario in the input scenario file by communicating with Planetlab nodes. The scenario file tells time, node and action for each event in the simulation. There are mainly three different

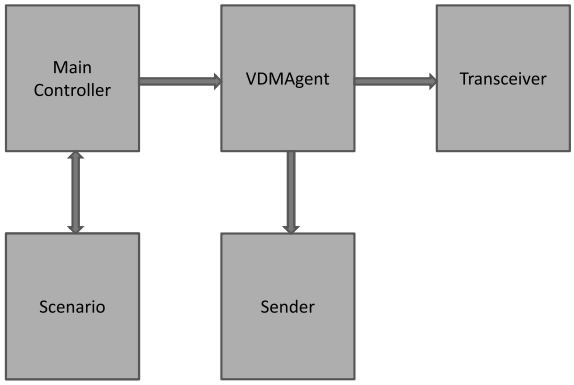

Fig. 17. Main components of PlanetLab implementation.

control messages from the main controller to the nodes. The first one is a connect message which alerts the node to connect to the multicast session. Another one is the disconnect message that tells the node to leave the session. The last message is terminate which is sent to every node at the end of the session.

A. Experiment Setup

For this evaluation, we used nodes only in the United States. We have identified a pool of working nodes that has around 140 nodes. Each time we select 100 nodes from this pool and run our experiment. We selected a node in Colorado as the source. When we ran more than one experiment on the same nodes at the same time, the performance got affected. So, we ran experiments one by one. An experiment is taking 5,000 s which is like a real session length since we are actually transmit-

ting a streaming traffic over the multicast tree. First 2,000 s are spent for join processes only. In the remaining 3,000 s, churn takes place. The reality that we have to run each experiment separately prevented us to run simulations many times. Each experiment is run 5 times with different seeds. We show average values of these 5 runs.

In PlanetLab implementation, we defined two more metrics in addition to the previously defined one.

- *Startup time:* When a node receives a connect command, it marks the time. When it is able to find a parent to connect and establish the connection it checks time again. The difference between two clock read is recorded as startup time for this node.
- *Reconnection time:* When a node receives a leave notification from its parent, it contacts its grandparent to rejoin the tree. We measure the time required for reconnection for the nodes whose parent leave.

B. Experiment Results

Fig. 18(a) shows average and maximum time needed for join process. Number of nodes that a new joining node has to query is increasing when number of nodes is increasing. This causes startup time to increase. We calculated average and maximum startup time. When number of nodes is 100, average time is around 0.4 s. The maximum time is 1.5 s. These values are reasonable to start receiving a stream. Maximum values are not related directly to number of nodes. Some nodes are responding late to ping request which causes startup time to increase.

Reconnection time is not related to number of nodes since orphan nodes start reconnection at their grandparent. Fig. 5(b) presents reconnection time versus number of nodes. Average time is around 0.2 s. There is no dependency on number of nodes. This 0.2 s interruption is experienced as jitter by user if there is no buffer.

In Fig. 18(c), we present results that give stretch values versus number of nodes. We show four different measurements. In some cases, nodes might have shorter path length to source whey they use overlay routing. The bottom line shows average stretch values of these nodes. The value is around 0.9. It means that these nodes receives stream with less delay than direct connection to source. We show overall average stretch value for all nodes. We also show average stretch values only for leaf nodes. Leaf in tree structure can be considered as worst places to be. Leaf nodes are expected long path length. Almost half of the nodes are expected to be at the leaf position in the tree. When we look at the average stretch for these nodes, it is a little higher than the general average. We also look at maximum stretch values. It goes up to 3 when number of nodes is 100.

In Fig. 18(d), we show hopcount versus number of nodes. Hopcount should increase with number of nodes. This increase depends on node degree and proportional to log of number of nodes. Average value for all tree is around 4. If we look at only leaf nodes, it is around 5 for number of nodes 100.

From Fig. 18(e), loss rate is increasing with number of nodes. In this experiment, we keep churn rate same while increasing number of nodes. So, number of joins and leaves is getting higher. When number of nodes is high, even though reconnection time doesn't change, number of nodes affected from a disconnection is high. This causes an increase in loss rate.

Overhead in Fig. 18(f) is increasing with number of nodes. Number of nodes that a node needs to contact for a join process is increasing. This causes an increase in overhead.

VII. SUMMARY AND FUTURE WORK

In this paper, we proposed a new overlay multicast protocol, VDM, that uses directionality among nodes to construct the multicast tree. By using the concept of directionality, VDM attempts to build its overlay tree congruent to the underlying network so that network resources are utilized efficiently while satisfying end-users in terms of perceived quality. We analyzed some corner cases one by one. Then we investigated complexity of join process.

Simulation results showed that VDM achieves better performance compared to HMTP in terms of various metrics like stretch, loss and overhead. VDM improved the path stretch which affects both overlay tree participants and physical network. Another improved metric is packet loss that is important for applications with real-time and/or reliable. We also showed that VDM causes less messaging overhead, that is a key factor for scalability of overlay multicast applications.

We also proposed a method to generalize the virtual distance between overlay nodes for automatically calculating overlay trees custom to specific application performance targets. By using the generalization concept, we aim to build target specific overlay trees that provide the ability to improve user perceived quality for specific purposes. We used two different metrics, delay and loss, for calculating the virtual distances and experimented with two version of VDM: VDM-D and VDM-L. Simulation results showed that VDM-D achieves better performance in terms of path length and stress while degrading the loss rate. On the other hand, VDM-L improves the loss rate performance, as expected, by sacrificing from stress and stretch.

We also have implemented our protocol on PlanetLab which gave us the opportunity to see real time characteristics of it. We showed some statistics such as startup time, reconnection time and stretch on PlanetLab.

The key insight of our work is that directionality may not achieve the best, but allows finding a good and practical tradeoff between quality expectations and protocol overhead. Integration of directionality with other approaches involving more special treatment of users quality expectations can be tried as a future work. Integration of super-nodes and multi-tree designs with VDM is a nice future work direction to take.

In reality most peers put a degree limit via their client software. No peer wants to undertake the burden of other peers – this is the typical behavior of a regular peer. We considered this behavior via assigning a degree_limit for every peer. However, super-peers will be different as they are more willing to take the burden of others. Future work should explore a variety of degree_limit patterns across the peers.

Another promising aspect to explore is to consider directionality in more than 1 dimension. While extending VDM's directional categorization of possible cases to two or more dimensional spaces, keeping the protocol's complexity at a practical level will be the challenge. However, it may prove worthwhile

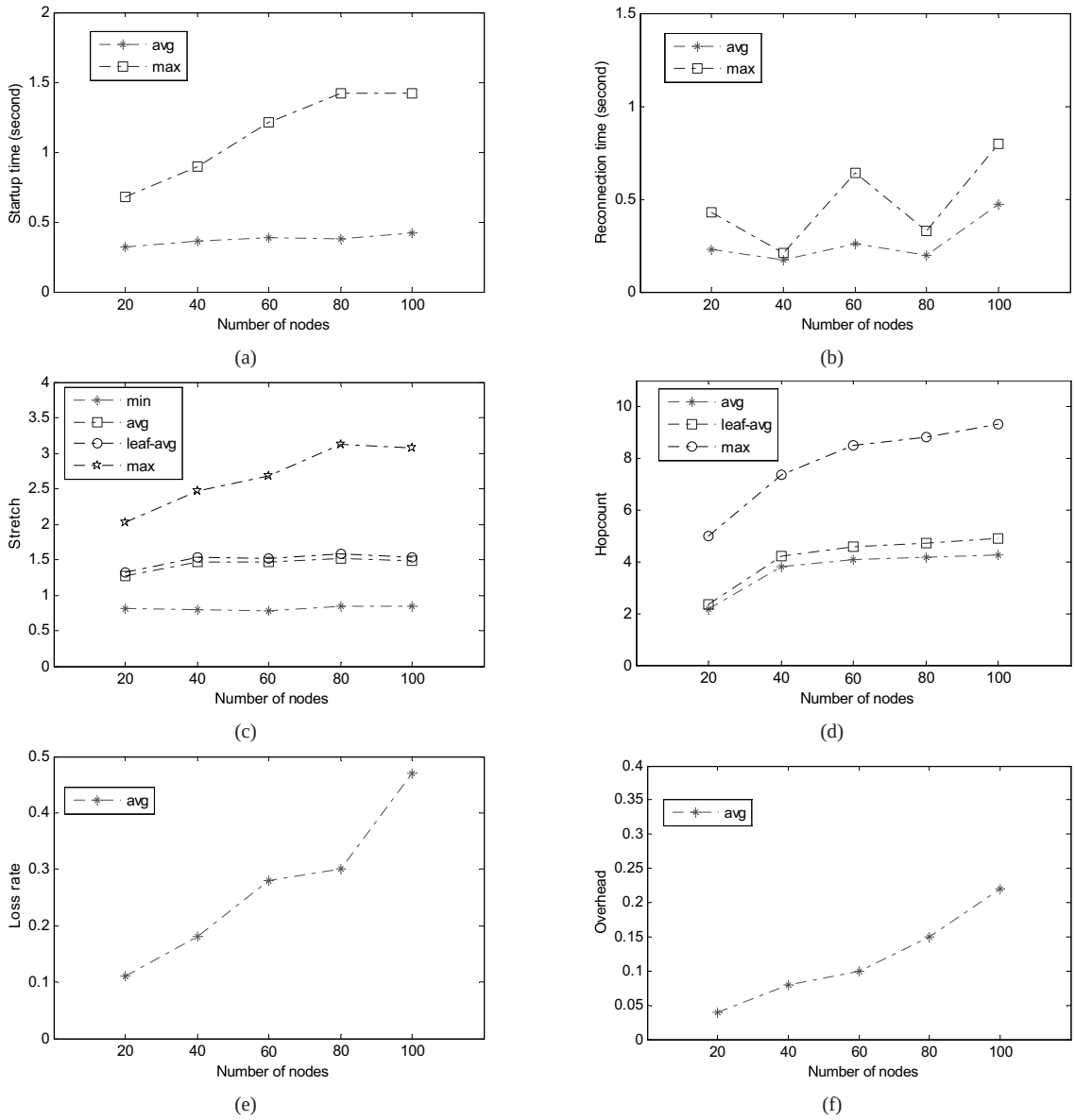

Fig. 18. VDM on PlanetLab: (a) Startup vs. number of nodes, (b) reconnection vs. number of nodes, (c) stretch vs. number of nodes, (d) hopcount vs. number of nodes, (e) loss rate vs. number of nodes, and (f) overhead vs. number of nodes.

to explore the concept since there are increasingly more multi-homed nodes on the Internet and 1-D directionality may become limited in the long run.

Adapting the overlay tree construction to targeted application metrics via the concept of generalized virtual distances deserve further investigation. We showed that VDM adapts its overlay tree for delay or loss by means of virtualizing the underlying network based on delay or loss on the links. It will be interesting to see how VDM performs when other link quality metrics are virtualized as "virtual distances", e.g., bandwidth, jitter, maximum delay, maximum loss, and reliability.

Last but not the least, in wireless networks, the constraints are different than wired networks. Although it may not be practical to directly apply VDM to the wireless networks, the general idea can give insights and a modified version can be used for overlay multicast construction in wireless networks. Since the challenges in wireless networks are energy scarcity, low computation power and high mobility, reducing the overhead of VDM against churn and dynamism will be of crucial importance. In our algorithm, we start the join process from the source node because that is the only one that we know. However, in wireless, the new node and the source node may not be within each other's coverage areas. One approach could be to use the wireless signal strengths, which are readily available via wireless transceivers, to estimate the distance between nodes. After modifying this join procedure, we might apply the directionality concept locally to converge towards MST. One of the reasons that we are using 1D is avoiding sibling communication. For omni-directional wireless, the 1D or 2D virtual directionality may prove to be useful to construct a relative orientation within a neighborhood of wireless nodes. With the increasing number of directional radio transceivers, this orientation could be improved.

REFERENCES

[1] S. Mercan and M. Yuksel, "Virtual distance: A generalized metric for overlay tree construction," in *Proc. IEEE ISCC*, July 2012, pp. 193–198.
[2] S. Mercan and M. Yuksel, "Virtual direction multicast for overlay networks," in *Proc. IEEE IPDPSW*, May 2011, pp. 1595–1601.
[3] IPTV News, [Online]. Available: http://www.iptvnews.net
[4] A. Mahimkar *et al.*, "Towards automated performance diagnosis in a large IPTV network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 231-242, 2009.
[5] M. Cha, W. A. Chaovalitwongse, Z. Ge, J. Yates, and S. Moon, "Path protection routing with SRLG constraints to support IPTV in WDM mesh networks," in *Proc. IEEE INFOCOM*, 2006.
[6] P2PTV, [Online]. Available: http://en.wikipedia.org/wiki/P2PTV
[7] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo, "P2P-TV systems under adverse network conditions: A measurement study," in *Proc. IEEE INFOCOM*, 2009.
[8] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, 1990.
[9] P. Francis, "Yoid: Extending the multicast Internet architecture," white paper, 1999.
[10] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 1, 2000.
[11] D. Pendarakis, S. Shi, D. Verma, and M.Waldvogel, "ALMI: An application level multicast infrastructure," *USITS*, vol. 1, 2001.
[12] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1489–1499, 2002.
[13] D. Helder and S. Jamin, "End-host multicast communication using switch-trees protocols," in *Proc. IEEE/ACM CCGRID*, 2002.
[14] Beichuan Zhang, Sugih Jamin, and Lixia Zhang "Host multicast: A framework for delivering multicast to end users," in *Proc. IEEE INFOCOM*, 2002.
[15] M. Castro *et al.*, "SplitStream: High-bandwidth multicast in cooperative environments," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, 2003.
[16] V. Venkataraman, P. Francis, and J. Calandrino, "Chunkyspread: Multi-tree unstructured peer-to-peer multicast," in *Proc. IPTPS*, 2006.
[17] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with delaunay triangulation overlays," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1472–1488, 2002.
[18] Y. Chawathe, S. McCanne, and E. A. Brewer, "RMX: Reliable multicast for heterogeneous networks," in *Proc. IEEE INFOCOM*, 2000.
[19] X. Zhang, X. Li, W. Luo, and B. Yan, "An application layer multicast approach based on topology-aware clustering," *Comput. Commun.*, vol. 32, no. 6, pp. 1095–1103, 2009.
[20] B. Cheng, M. Yuksel, and S. Kalyanaraman, "Virtual direction routing for overlay networks," in *Proc. IEEE P2P*, 2009.
[21] M. Bishop and S. Rao, "Considering priority in overlay multicast protocols under heterogeneous environments," in *Proc. IEEE INFOCOM*, 2006.
[22] X. Zhang, J. Liu, B. Li, and Y. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM*, 2005.
[23] S.W. Tan, A. Waters, and J. Crawford, "Meshtree:A delay- optimised overlay multicast tree building protocol," in *Proc. ICPADS*, 2005.
[24] M. Kwonand and S.Fahmy, "Path-aware overlay multicast," *Comput. Netw.*, vol. 47, no. 1, pp. 23–45, 2005.
[25] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Omni: An efficient overlay multicast infrastructure for real-time applications," *Comput. Netw.*, vol. 50, no. 6, pp. 826–841, 2006.
[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: W. H. Freeman, 1979.
[27] The network simulator — ns-2, [Online]. Available: http://www.isi.edu/nsnam/ns/
[28] Gt-itm: Georgia tech internetwork topology models, [Online]. Available: http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html
[29] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proc. ACM SIGCOMM*, 2006.
[30] O. Herrera and T. Znati, "Modeling churn in p2p networks," in *Proc. IEEE ANSS*, 2007.
[31] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 377–390, 2009.
[32] T. Ng and H. Zhang, "A network positioning system for the Internet," in *Proc. USENIX*, 2004.
[33] L. Tang and M. Crovella, "Virtual landmarks for the Internet," in *Proc. ACM SIGCOMM*, 2003.
[34] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "VVivaldi: A decentralized network coordinate system," in *Proc. ACM SIGCOMM*, 2004.
[35] M. Costa, M. Castro, R. Rowstron, and P. Key, "PIC: practical Internet coordinates for distance estimation," in *Proc. ICDCS*, 2004.
[36] [Online]. Available: http://www.akamai.com/html/technology/dataviz2.html
[37] [Online]. Available: http://iplane.cs.washington.edu/
[38] [Online]. Available: http://www.planet-lab.org

Suat Mercan is Assistant Professor at Zirve University in Turkey. He received Ph.D. degree in Computer Science at University of Nevada-Reno (UNR), Reno, NV in 2011. He received his M.S degree in Electrical and Computer Engineering from University of South Alabama (USA), Mobile, AL in 2007. He received his B.S. degree from Computer Engineering from Fatih University, Istanbul, Turkey in 2005. His main research interests are peer-to-peer networks, multicasting in overlay networks, IPTV, content delivery. He is a Member of IEEE.

Murat Yuksel is currently an Associate Professor at the CSE Department of The University of Nevada - Reno (UNR), Reno, NV. He was with the ECSE Department of Rensselaer Polytechnic Institute (RPI), Troy, NY as a Postdoctoral Research Associate and a member of Adjunct Faculty until 2006. He received a B.S. degree from Computer Engineering Department of Ege University, Izmir, Turkey in 1996. He received M.S. and Ph.D. degrees from Computer Science Department of RPI in 1999 and 2002 respectively. His research interests are in the area of computer communication networks with a focus on protocol design, network economics, wireless routing, free-space-optical mobile ad-hoc networks (FSO-MANETs), and peer-to-peer. He is a Member of IEEE, ACM, Sigma Xi, and ASEE.