Noname manuscript No.
(will be inserted by the editor)

On-Demand Waypoints for Live P2P Video Broadcasting

Aditya Ganjam, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan and Hui Zhang

Abstract A peer-to-peer architecture has emerged as a promising approach to enabling the ubiquitous deployment of live video broadcasting on the Internet. However the performance in these architectures is unpredictable and fundamentally constrained by the characteristics of the members participating in the broadcast. By characteristics, we refer to user dynamics, out-going bandwidth connectivity, whether the member is behind NAT/firewall, and the network conditions among participating members. While several researchers have looked at hybrid P2P/CDN approaches to address these issues, such approaches require provisioning of centralized server resources prior to a broadcast, which complicates the goal of *ubiquitous* video broadcasting. In this paper, we explore an alternative architecture where users are willing to donate their bandwidth resources to a broadcast event, even though they are not a participant in the event. Such users constitute what we term a *waypoint* community. Any given broadcast event involves constructing overlays only based on participants to the extent possible, however waypoints may be dynamically invoked in an on-demand, performance-

A. Ganjam
School of Computer Science
Carnegie Mellon University
E-mail: aganjam@cs.cmu.edu

S. Rao
School of Electrical and Computer Engineering
Purdue University
465 Northwestern Avenue, West Lafayette, IN 47907, USA
Tel.:+1-765-494-3399
Fax: +1-765-494-0676
E-mail: sanjay@purdue.edu

K. Sripanidkulchai
IBM T.J. Watson Research Center
E-mail: kunwadee@us.ibm.com

J. Zhan
School of Computer Science
Carnegie Mellon University
E-mail: jibin@cs.cmu.edu

H. Zhang
School of Computer Science
Carnegie Mellon University
E-mail: hzhang@cs.cmu.edu

driven fashion to improve the performance of a broadcast. We present the design of a system built on this idea. Detailed results from trace-driven experiments over the PlanetLab distributed infrastructure and Emulab demonstrate the potential of the waypoint architecture to improve the performance of purely P2P-based overlays.

1 Introduction

In the last few years, a peer-to-peer approach has emerged as a key alternative to enabling video broadcasting applications on the Internet [2, 6–8, 11, 17, 19, 26, 28, 38]. The key aspect that makes such an approach attractive is that a participant that tunes into a broadcast is not only downloading a video stream, but also uploading it to other participants watching the stream. Consequently, no dedicated infrastructure is required, deployment is instantaneous, and the system can scale to large group sizes because greater demand also generates more bandwidth resources.

While a peer-to-peer architecture has promise, the feasibility and performance with the approach is intrinsically dependent on the constitution of members participating in the broadcast. In many real deployment settings, a large number of participating members are behind asymmetric connections (download $\gg$ upload) such as DSL, and there is insufficient bandwidth resources to ensure all members receive the full source rate [7, 33]. Further, up to 80% of these members may be behind NATs and firewalls, severely limiting the connectivity in the system [12]. In addition, the performance achievable is dependent on the dynamicity of participating members, and dealing with congestion and the variable quality of paths on the Internet is a challenge.

To handle these limitations of a purely peer-to-peer architecture, several researchers have looked at hybrid architectures involving CDNs and peer-to-peer systems [3, 15, 16, 18, 36]. These designs rely on dedicated and strategically placed servers that must be provisioned with large bandwidth capabilities prior to the start of a broadcast, with P2P resources being used to minimize the overhead at servers. While such dedicated infrastructure support is feasible with high-end content providers, it complicates the goal of achieving ubiquitous video broadcasting, i.e. allowing any participant on the Internet to launch a broadcast.

In this paper, we explore an alternative architecture that does not involve static pre-provisioned infrastructure support. Instead, we envision a model where users organize into what we term *waypoint* communities, formed around interests of users. For example, music fans may donate their hosts to form a waypoint community that is available for peer-to-peer based radio stations and live music concerts. The key characteristic of a waypoint community is that members of the community are willing to donate their bandwidth resources to any broadcast event in the community, even though they are not a participant in the event.

Our waypoint model must be distinguished from the use of "seeds" in BitTorrent [9]. Like waypoints, seeds in BitTorrent also contribute bandwidth even though they are not themselves downloading. However, the crucial difference in our waypoint model is that we are dynamically invoking waypoints when the bandwidth resources of the environment is deemed inadequate, and releasing them when plentiful. Second, given the live nature of video streaming, waypoints too are receiving content and consuming bandwidth resources with the rest of the participants, unlike seeds in BitTorrent that have already downloaded content. Thus careful selection of waypoints becomes important, and poor selection could hurt the system.

We present the design of a prototype system to realize the waypoint community model for live streaming. Participating members are used to forward traffic to the extent possible

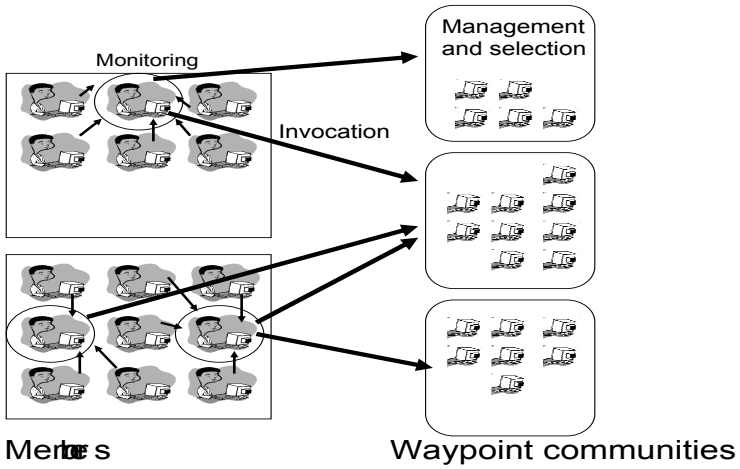


Fig. 1 Waypoint architecture.

and waypoints can be used to provide more bandwidth and stability. The performance of the group is monitored in a fully distributed fashion, and a significant decrease in the performance of members in the system triggers an invocation of *waypoints*. At a later point, if waypoints are no longer needed, then they are released from the application. Given that waypoints are themselves hosts of unknown performance, our system includes mechanisms to rate waypoints and select the better-performing ones. This *on-demand* and *performance-driven* invocation of waypoints distinguishes our approach from both hybrid P2P/CDN systems, and from the use of seeds in file-download systems like BitTorrent.

We have implemented our waypoint system to work with an operationally deployed overlay based broadcasting system, though the design is general so that it can easily be integrated with any other broadcasting system or protocol as well. We present evaluation results on Emulab, and on Planetlab using emulations of traces from real broadcasting events. Our results show that our system can significantly improve performance when used in conjunction with the base broadcasting system. The performance benefits stem both from the additional resources, as well as the stability that waypoints provide.

The rest of the paper is organized as follows. Section 2 presents an overview of the waypoint architecture. Section 3 presents the system design. The evaluation methodology is presented in Section 4, and Section 5 presents the results.

2 Waypoint Architecture Overview

In this section, we discuss key issues that impact the feasibility of a waypoint architecture. We then discuss the main architectural components of the waypoint model.

2.1 Architectural Feasibility

The utility of a waypoint architecture hinges on two properties: (i) quality of waypoint resources, and (ii) willingness of hosts to become waypoints. If the quality of waypoints

resources is poor or few people are willing to become waypoints, then waypoints cannot improve the performance of a broadcast.

In terms of quality of resources, we expect waypoint resources to be as heterogeneous as application end-point resources. However, with waypoints, we have control over when waypoints can participate in a broadcast and can afford to be more selective. For example, only waypoints that have good quality resources and can provide good performance are selected to help in a broadcast. If a waypoint does not provide good performance, we can remove the waypoint from the broadcast. Thus, a key part of the waypoint architecture involves algorithms to select and rate waypoints, so only good waypoints may be retained.

To get a sense of the potential of the waypoint model, we have analyzed the bandwidth distribution for nodes participating in the largest broadcast event from the Akamai content delivery network where the peak group size was 74,000 nodes [33]. Our results indicate that if any of the nodes were to be used as waypoints for this event when they were not tuning in, the amount of bandwidth resources could increase by at least an additional 40% assuming each host could contribute up to 6 times the source rate, and by 120%, if each host could contribute upto 20 times the source rate. These results indicate that the use of waypoints can significantly increase the feasibility of the overall system.

The way in which waypoint communities are formed can influence the willingness of people to contribute resources. For example, one may rely on altruism similar to how hosts participate in SETI@home [31] to contribute idle computation resources or peer-to-peer file-sharing applications to contribute storage and bandwidth. An alternative is to provide a pay-per-use model where waypoints have an incentive to contribute as they are directly compensated for their contributions. Another promising model is to form waypoint communities around the interests of users. People are more motivated to contribute to their community when there is a strong sense of shared purpose or interest. For instance, one could imagine a waypoint community involving rock music, or a community used to broadcast educational content, where the waypoint hosts are used only for broadcast events with that type of content. Furthermore, as compensation, they receive social recognition within their community for their contributions.

Yet another approach is to reward nodes for their contribution to the peer-to-peer community through better performance in the broadcasts they are part of. Two components are needed to achieve this. First, distributed rating mechanisms are required to rate peers for their contributions. Second, this information must be used to improve the performance of peers in their broadcast. To solve the first problem, several recent research proposals for the distributed auditing and rating of nodes [5, 10, 20] can be leveraged. In particular, one solution based on [5], could involve all members in the community being part of a Distributed Hash Table (DHT), with the ratings maintained using the DHT. Achieving differential performance of peers in the broadcast is an active area of research, and a number of recent proposals such as [1, 4, 21, 23, 24, 29, 32, 34, 37] can be leveraged. In one relevant proposal [34] targeted at regimes where not all peers in a broadcast can receive the full bandwidth, the amount of bandwidth a receiver is actually entailed to depends on the total contribution that it makes, thus ensuring nodes that contribute more receive better quality. In another proposal [4], peers that contribute more are placed at more desirable locations in the overlay structure. For instance, peers may be rewarded by being placed closer to the root in tree-based structures, thereby ensuring they see fewer disruptions due to peer departures upstream, and get better quality. Thus, given the rich body of existing work on incentives that can be leveraged by the waypoint architecture, in this paper, we do not further consider incentive mechanisms.

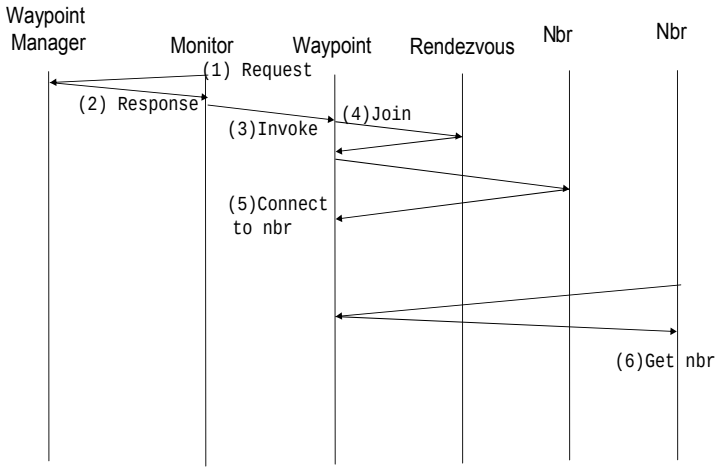


Fig. 2 Sequence of actions in waypoint invocation.

2.2 Architectural Components

To realize the waypoint architecture, three top-level components are required, as illustrated in Figure 1: monitoring, waypoint invocation, and waypoint management.

Monitoring: This component dynamically monitors the bandwidth resource and performance information of members during the broadcast. This information is used to determine if and when waypoints are needed. In our architecture, the monitors depicted with circles in the left-hand side of Figure 1 are group members that perform the additional functionality of aggregating information. As the group size grows, more members become monitors.

Waypoint invocation: Monitors execute an invocation decision algorithm based on the collected information. The algorithm decides when to invoke and remove waypoints, and determines the number of waypoints that need to be invoked and the waypoints' locations. Monitors contact a set of waypoint managers when waypoints are needed. Upon receiving the waypoint list, monitors invoke them to join the broadcast. Waypoints are removed when they are no longer needed or when their performance is poor.

Waypoint Management and Selection: When waypoints are requested by a monitor, the manager selects an appropriate set of waypoints to return. The selection algorithm may optimize for various metrics such as closest set of waypoints or smallest set of waypoints. To facilitate the selection, waypoint managers keep track of waypoints in their communities and their associated properties such as waypoints' geographic or network locations [25] and waypoints' available upstream and downstream bandwidth. The algorithm uses history of previous performance of waypoints to decide whether or not they are useful.

Putting everything together, Figure 2 depicts the sequence of action involved in waypoint invocation. First, the monitor sends a waypoint request message to the waypoint manager denoted as (1) in the figure. The waypoint manager responds with a list of potential waypoints (2). The monitor contacts the individual waypoints and asks them to join the broadcast (3). Each waypoint then joins the broadcast using the same steps as a typical group member. The waypoint initially contacts the rendezvous point to get a list of members in the group (4). It then determines the appropriate neighbors in the overlay, as per the standard mechanisms of the underlying protocol.

3 System Design

In designing the system we have the following goals:
• *On-demand, performance-driven invocation and removal of waypoints:* Because of high churn rate and heterogeneity of participating members, it is difficult to statically provision waypoints apriori. The system should dynamically invoke waypoints when and where waypoints can help to improve the performance of members. The system must make efficient use of waypoints, and only invoke as many waypoints as needed to have good performance.
• *Robustness to poor waypoints:* Since waypoints are also end hosts and not necessarily well-provisioned or dedicated machines, the systems should monitor the performance of waypoints and remove poor waypoints.
• *Minimize dependence on overlay protocol:* The waypoint framework has generic components which can be applied across different P2P streaming protocols [6–8, 11, 16, 17, 19, 22, 26, 27, 38], and perhaps more protocol-specific components for performance optimization if needed. The generic part of the framework consists of the information needed from the P2P streaming protocol about individual nodes such as node performance and resource availability.

3.1 Waypoint Invocation Decision

In this section, we discuss the step prior to waypoint invocation where monitors determine if waypoints are needed, how many and where. As a first-cut approach, we could invoke waypoints when some group members see poor performance. However, this is a bit simplistic as we need to consider two other factors in addition to performance. First, if certain members have poor performance because of their own last mile bottleneck making them incapable of receiving the full source rate then invoking waypoints to help these members will not help. In addition, looking only at performance does not provide any insight into when we can release waypoints.

To address these issues, we introduce another parameter that considers the properties of the underlying environment in addition to the current performance that nodes are seeing. We capture the environment using the *Resource Index (RI)* metric defined as the ratio of the number of nodes that could be *potentially supported* to the number of nodes currently in the system. for a particular source rate [7]. An *RI* of 1 indicates that the system is saturated, and an *RI* of less than 1 indicates that not all the nodes can receive the full source rate. As the *RI* gets higher, the environment becomes less constrained and it becomes more feasible to construct a good overlay structure. We note that the *RI* is impacted both by the upstream bandwidth capabilities of nodes, and the connectivity constraints imposed by the presence of nodes behind NATs and firewalls.

Our waypoint invocation decision combines both the current performance nodes see and the *RI* by iteratively monitoring, setting target performance and *RI* for the system, and moving the system to operate at target by adding or removing waypoints.

3.1.1 Monitoring operational conditions

The monitors periodically collect the following information from each node (member or waypoint): (i) the bandwidth that the node is receiving (R_{BW}); and (ii) the residual upstream bandwidth that the node can contribute (AS_{BW}). This information is used to compute two operational values:

• *Current Delivery Ratio (CDR):* This is the ratio of the bandwidth a node receives to the streaming source rate, averaged across all the nodes.
• *Operational RI (ORI):* This is the current *RI* at which the system is operating, including all nodes (members and waypoints). If $P(t)$ is the set of nodes with public IP addresses and $R(t)$ is the set of nodes behind NAT or firewall at time t then the *ORI* is:

$$\frac{\sum_{i \in (P(t),R(t))} R_{BW}[i] + \sum_{i \in P(t)} AS_{BW}[i]}{totalNodes * streamingSourceRate} \quad (1)$$

Intutively, the *ORI* measures the ratio of the total bandwidth supply in the system (numerator) to the total demand for bandwidth (denominator). We estimate the total bandwidth supply as the sum of (i) the bandwidth currently being received by all nodes; and (ii) the residual upstream bandwidth of public nodes. We only consider nodes with public IP addresses in term (ii), since nodes behind NATs and firewalls can only tap the supply bandwidth at public nodes.

3.1.2 Setting Targets

Next, we define target performance and resource index, and discuss how we estimate and set those values for a given broadcast.
• *Target Delivery Ratio (TDR):* This is the best possible value of *CDR* that the system can achieve at any point in time. Note that this ratio may be less than 1 due to the presence of nodes that have contrained last mile bandwidth.
• *Target RI (TRI):* The *RI* value that the system estimates it must operate at to ensure it can achieve the desired TDR.

Rather than setting static targets which may not capture the complexity of dynamic conditions during a broacast, we take an adaptive approach. We learn and estimate target values throughout the duration of the broadcast which we describe next. We also evaluate our approach in further detail in Section 5.1 particularly against a scheme that sets static targets. Our adaptive algorithm has two modes of operation: (i) calibrating target, and (ii) moving towards target.
Calibrating Target: If the system is performing below target ($CDR < TDR$), but we think that there are sufficient resources in the system ($ORI \geq TRI$), then we may have previously underestimated the TRI. We then add more waypoints and increase the TRI to the current value of ORI. In our implementation, waypoints are invoked only if the *CDR* is less than 0.95 of the *TDR*.

If repeated invocation of waypoints does not significantly improve the performance, then we may have previously overestimated the target performance because there are more bandwidth constrained nodes currently in the system. We dampen the waypoint invocation by reducing TDR and TRI to the current operational values.

On the other hand, if the system is performing better than it was previously estimated to be capable of ($CDR > TDR$), perhaps because of a reduction in the number of bandwidth constrained nodes, then TDR and TRI are reset to the current values. Likewise, the values are reset if the system is observed to provide target performance at a lower resource index than estimated ($CDR = TDR$ && $ORI < TRI$).
Moving Towards Target: If both operational values are lower than target ($ORI < TRI$ && $CDR < TDR$), an appropriate number of waypoints is *invoked* to ensure the system operates closer to the TRI. If the current performance meets target (CDR is approximately equal to the TDR), but there is an excess of resources in the system ($ORI > TRI$) then

waypoints are *removed* from the system to move the ORI closer to target. To avoid oscillations between adding and removing waypoints, we use a hysteresis.

In our implementation, we make invocation decisions every *invocation cycle period*. The period should reflect the expected time from the decision to invoke waypoints to the point where the waypoint is helping members of the group. We set this period to 20 seconds in our experiments.

3.2 Group Monitoring

In our system, a subset of the participating members serve as the monitors and decide when waypoints must be invoked, and released. For monitoring, we introduce a separate structure decoupled from the overlay data forwarding structure. For scalability, the monitoring system is organized using a two-level hierarchy. Every member belongs to a cluster. A cluster head which is also a member participating in the broadcast, monitors the cluster members and invokes and removes waypoints for the cluster. Cluster heads are elected by members of the cluster using a distributed leader election protocol.

A monitoring rendezvous point keeps track of the current list of clusters, and the heads of each cluster. When a member joins the broadcast, it gets a list of clusters from the rendezvous point. In addition to connecting to the data delivery structure, it also connects to the monitoring structure by selecting a cluster head to whom it will report. The primary criteria in choosing a cluster head is proximity to the head measured through a combination of latitude/longitude, and GNP coordinates [25].

Members periodically report their bandwidth resource and performance information to the their cluster head. This information is used to invoke and remove waypoints as discussed in Section 3.1. The information collected for each member includes the actual bandwidth received by the member (computed using a window of two seconds in the implementation), the total amount of bandwidth it can forward at this time (supply bandwidth), the amount of bandwidth it has available to forward to new children (available supply bandwidth), and whether the member is behind a NAT/firewall.

In order to ensure the sizes of clusters are maintained reasonable, clusters that are too large may be split. The cluster head selects a subset of participants in the cluster to form a new cluster, decides on one of them as the head of the new cluster, and informs the selected participants of the new cluster head. Likewise, when the size of a cluster becomes small, the cluster may merge with another larger cluster. The cluster head decides which cluster to merge with and informs all members of its cluster to join the new head. The cluster heads are normal group members and may leave the system. In the case of a graceful leave, the cluster head simply picks a new head, and informs all members about the new head. To handle abrupt death, a distributed leader election algorithm based on Gossip is employed [14].

3.3 Waypoint Selection, Bootstrap, and Rating

Having decided to invoke waypoints, the monitors contact the waypoint managers for a set of waypoints. The request includes the location of the monitor in terms of latitude/longitude or virtual network coordinates like GNP [25], and the total additional bandwidth supply needed.

Upon receiving a request, a waypoint manager selects a set of waypoints based on the potential bandwidth that each waypoint can supply, and the proximity between the way-

point and the requesting monitor. Waypoints are prioritized based on their contribution, and among waypoints with similar contributions waypoints located in proximity to the monitor are preferred. It should be noted that in meeting the total bandwidth requirement, the waypoint manager must also consider the bandwidth to be consumed by each waypoint.

The monitor sends each selected waypoint a join message with the required configuration to join the event. Minimally, this information includes the IP address and port of the rendezvous point. We also look at an optimized method we term *WaypointHint* in which an invoked waypoint is provided with a hint regarding which participating members are seeing poor performance. After the waypoint successfully connects to the overlay, it sends messages to each member in the list telling them of its presence. Members in the list with poor performance can select the waypoint as its neighbor.

Waypoints are not provisioned servers, and may themselves provide poor performance. Consequently, our system needs to be selective and use only good waypoints. To facilitate this, monitors track the total bandwidth contributed by each invoked waypoint, and may remove a waypoint if the contribution does not exceed the source rate by a certain minimum threshold. A waypoint may also be removed if its average contribution to each of its neighbors is less than a certain threshold.

A monitor that invokes a waypoint reports to the waypoint manager on the effectiveness of the waypoint, including the average contribution provided. The waypoint manager maintains a history of each waypoint's performance, which may be used to guide future invocation decisions. If a waypoint has repeatedly not provided adequate performance improvements, it is less likely to be invoked in the future.

3.4 Protocol specific customizations

Much of the waypoint system described so far is generic and independent of whether the underlying P2P streaming protocol is tree-based [6–8, 11, 17, 26], or mesh-based [16, 19, 22, 27, 38]. We next discuss additional heuristics specific to tree-based protocols that can help ensure better performance with these protocols. In a tree-based protocol, when a node upstream on the overlay has poor performance, all nodes downstream also have poor performance. It is very likely that we only need to invoke one waypoint for the node upstream that is the root cause of the problem as opposed to invoking waypoints for the node upstream and all nodes downstream. To account for such cases, we introduce another measure called the *last hop delivery ratio* which is defined as the bandwidth received at the node over the bandwidth sent by the parent. Using the last hop delivery ratio as a filter, only the nodes that have a last hop delivery ratio of less than 1 (i.e., the root cause of the problem) are considered when computing the number of waypoints to invoke. Likewise, in order to minimize disruption to system performance with tree-based protocols, a waypoint is removed only if the number of descendants it has does not exceed a threshold. Further, waypoints with the fewest descendants are removed first to minimize the impact on performance.

4 Evaluation

We have integrated our waypoint system with an operationally deployed broadcasting system [7] that has support for important practical aspects such as NATs/firewalls, and heterogeneous node capabilities. We expect it would be relatively straightforward to integrate our waypoint system with other overlay broadcasting systems as well.

	Entire Trace	Trace 1	Trace 2
Mean Session Duration	18 min	5 min	6 min
Mean Interarrival time	17 sec	5 sec	7 sec
% of members that can stream less than src rate	76%	71%	78%
% of members behind NAT/Firewall	70%	74%	82%

Table 1 Characteristics of participants in the entire *Slashdot* trace and in two 1000 second snapshots.

The system employs an overlay protocol which constructs a tree for data delivery. The tree is primarily optimized for bandwidth and secondarily for delay. Members maintain knowledge about other group members using a gossip-like algorithm. Members monitor their performance in the tree - if this is not found satisfactory, they probe other members, and choose a new parent based on delay, performance, and available bandwidth of probed candidates.

The goal of our evaluation is to answer the following:

• Is our system effective in enabling good application performance by invoking waypoints?

• Is our system invoking an appropriate number of waypoints? Enough waypoints must be invoked to provide good performance, but an excessive number is not desirable.

• Is the system intelligently invoking well-performing waypoints and ignoring poor-performing ones?

• How well does our system behave in realistic Internet environments? How well does our system work in a range of environments of different resource and dynamicity levels?

In the rest of this section, we present the performance metrics and our experimental methodology.

4.1 Performance Metrics

Our evaluations employ the following indices and metrics:

• *Application Performance:* We measure application performance by considering the average bandwidth receivers see during the session.

• *Operational resource index (ORI):* This captures the resource index of the system when considering all participating members *and* waypoints as previously defined in Section 3.1. This metric helps measure whether the system is invoking an appropriate number of waypoints. At least as many waypoints must be invoked to achieve an *ORI* of 1.0 but computing the optimal *ORI* is difficult, as participant dynamics, and congestion on Internet paths may prevent the use of bandwidth resources even if they are available.

• *Native Resource Index (NRI):* This index is similar to the *ORI*, but only considers members (i.e., no waypoints) in the system to capture the inherent availability of resources in the system without the help of waypoints.

• *Use of waypoints:* We consider the number of waypoints used in a run to measure the dependence on waypoints, and the *Supply Fraction*, or the fraction of total bandwidth in the group that is forwarded by waypoints.

4.2 Methodology

Our evaluations are conducted on Planetlab and Emulab [35]. The Planetlab experiments enable us to test our system under realistic Internet settings. The Emulab experiments are

complementary in that we can evaluate particular aspects of the system under a controlled environment.

We conducted experiments on Planetlab by emulating a trace from a real broadcast event [7]. We use one of the largest traces which was a broadcast conducted to the Slashdot community, that we refer to as *Slashdot*. In our discussions we refer to an incarnation as every instance of a unique user (as identified by its $< publicIP, privateIP >$ pair) that joins the broadcast.

This trace has a total of 1609 incarnations, and a peak size of 160 incarnations with characteristics summarized in Table 1. Members were found to exhibit a Pareto-like distribution of stay times, with the mean stay time being 18 minutes and a mean inter-arrival time of 17 seconds. About 76% of the incarnations were behind DSL/cable modem connections with insufficient upstream bandwidth to support the source streaming rate. About 24% were behind Ethernet connections with large upstream bandwidth. In addition, about 70% of the incarnations were behind NATs.

We emulate a trace on Planetlab, by having each incarnation in the trace run on a Planetlab host. Given that the peak number of incarnations in the trace are much larger than the number of Planetlab hosts, multiple simultaneously participating incarnations in the trace may be mapped onto the same Planetlab host. Depending on whether the incarnation in the trace is behind a DSL or Ethernet connection, we assume it contributes 0 or 6 times the source rate, consistent with [7]. We emulate NAT and firewall connectivity restrictions by implementing packet filtering to ensure that two Planetlab incarnations that are behind a NAT cannot communicate with each other. We directly use the same group dynamics pattern as in the trace to drive the experiment.

In order to ensure as realistic an emulation as possible, we map each incarnation in the trace to the geographically closest Planetlab host whose bandwidth constraints will not be violated. We use GeoBytes [13] to map an IP address to its geographic location in latitude and longitude coordinates. To ensure that we do not place an undue bandwidth demand on the Planetlab hosts, we require that the following invariant is maintained during the mapping process: $\sum_{i=1}^{j} B_j < B_{pli}$ where j is the number of incarnations in the trace mapped to a Planetlab host i, B_j is the maximum upstream bandwidth that the incarnation in the Slashdot trace contributes, and B_{pli} is the maximum upstream bandwidth of the Planetlab host i. We estimate the maximum upstream bandwidth of hosts in Planetlab using data from iPerf [30].

Our Emulab experiments are used to explore the performance of our schemes in environments with different levels of dynamicity. We do not emulate link delay or loss because the focus of our evaluation in this section is to measure the system performance with regard to group dynamics in isolation. The duration that members stay in the group are drawn from a Pareto distribution and the interarrival times are drawn from an exponential distribution, both of which are commonly observed distributions in video streaming systems [33]. We assume the out-going bandwidth supply of a member follows a distribution shown in Table 2 based on the largest event presented in [33]. We use a low bandwidth source rate of 20 kbps to keep the load in terms of bandwidth and packet count low on Emulab nodes and the network.

5 Results

In this section, we present evaluation results of our waypoint system. Section 5.1 evaluates the importance of an adaptive invocation scheme, and the potential benefits of waypoints in

Contribution (normalized to source rate)	0	1	2	5	6
Percentage of members	60.9%	18.7%	8.4%	5.2%	6.8%

Table 2 Member bandwidth contribution distribution normalized by source rate for the largest event presented in [11].

improving performance in environments with high churn. Section 5.2 evaluates the effectiveness of the waypoint invocation algorithm in enhancing performance in poor resource environments with low NRI. Section 5.3 evaluates how quickly our system can invoke waypoints to improve system performance. In Section 5.4, we evaluate the importance and effectiveness of the waypoint rating schemes in coping with variability in waypoint performance. Our experiments in Section 5.1 are conducted using Emulab, while all other experiments are conducted on Planetlab. Each result is the average of 3 runs.

5.1 Importance of Adaptive Targets for Invocation

In this section, we evaluate the importance of an adaptive invocation algorithm, and the need to dynamically determine the *TRI* as opposed to merely invoking waypoints to meet a static target resource index. Our evaluations also explore the benefits of waypoints in enhancing system stability.

We run experiments on Emulab with two different levels of dynamics. *Environment1* has a 30 minute stay time and 5 second interarrival time, while *Environment2* represents a more dynamic environment with a 10 minute mean stay time and 2 second interarrival time. We note that in both *Environment1* and *Environment2* the *NRI* is always above 1.0, indicating there is sufficient bandwidth resources among participating members for the system to be self-sustaining - thus waypoints are primarily invoked for their ability to enhance system stability.

Our experiments compare the performance of our system running two different versions of the waypoint invocation decision algorithm:

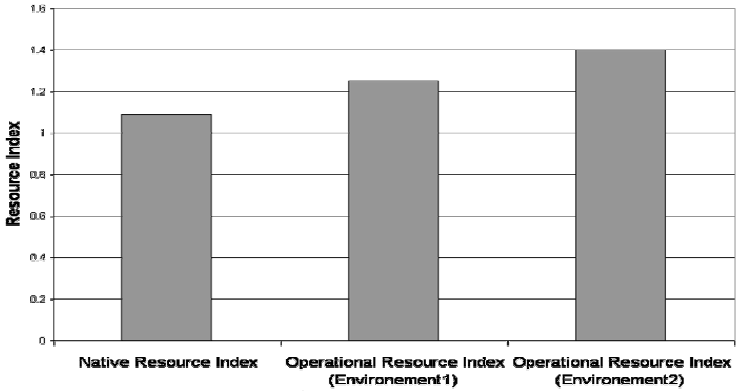
• *Adaptive algorithm:* This is the algorithm that adaptively sets targets (*TRI* and *TDR*) as discussed in Section 3.

• *Static-RI(t) :* We use this heuristic for baseline comparisons. The system monitors the overall resource index of nodes in the broadcast and invokes waypoints if the *ORI* is lower than a static threshold t and removes waypoints if the *ORI* is greater than $t + \delta$. In our experiments we set δ to 0.1.

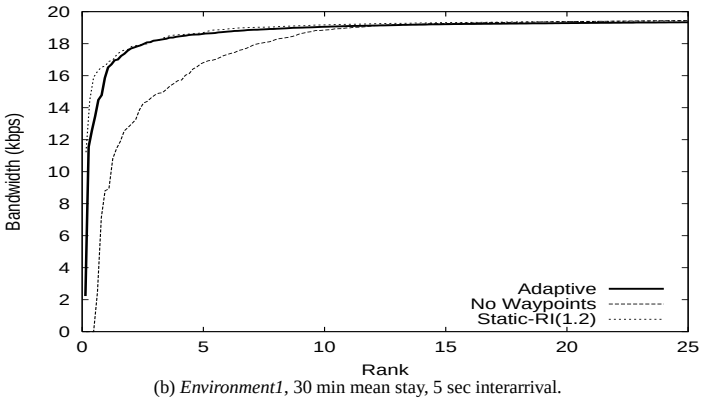
Figure 3(b) and Figure 3(c) plot the cumulative distribution of the mean bandwidth obtained by receivers in *Environment1*, and *Environment2* respectively when using various invocation algorithms. The Y-Axis corresponds to the bandwidth, and the X-Axis corresponds to the fraction of receivers that obtain less than a certain bandwidth. Only the worst performing 25% of the receivers are shown for clarity. Each curve corresponds to a different invocation algorithm, and curves to the left represent better performance.

In both environments, the performance without waypoints is not satisfactory. When the *Static-RI* schemes are considered, *Static-RI(1.2)* was able to significantly improve performance in the low dynamic *Environment1*. However, it was insufficient in the more dynamic *Environment2*, where *Static-RI(1.5)* led to much better performance.

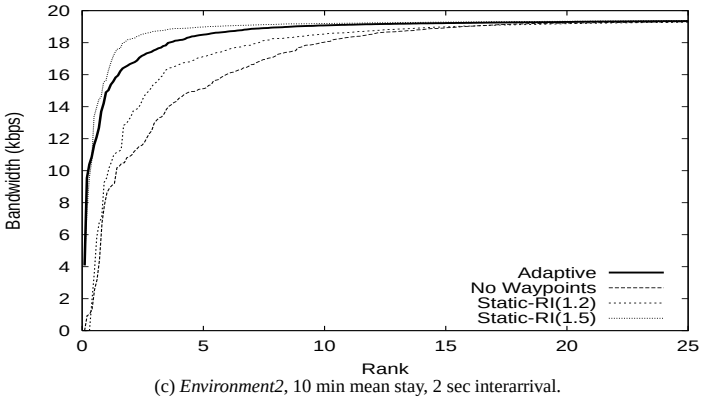
Figures 3(b) and 3(c) show that our *Adaptive* algorithm performs well in both environments. Its performance is comparable to *Static-RI(1.2)* in the lower dynamic environment



(a) *NRI* and *ORI* for *Environment1* and *Environment2*.



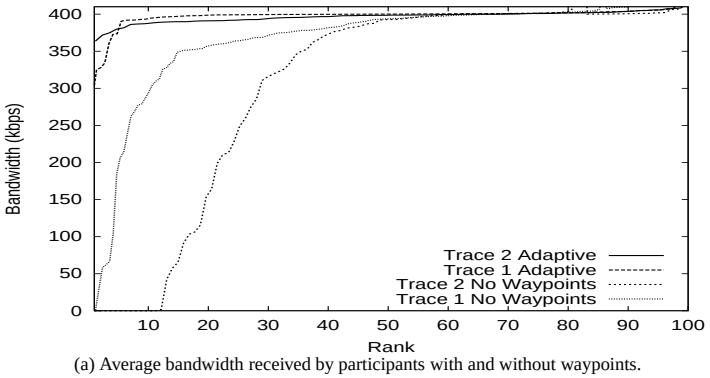
(b) *Environment1*, 30 min mean stay, 5 sec interarrival.



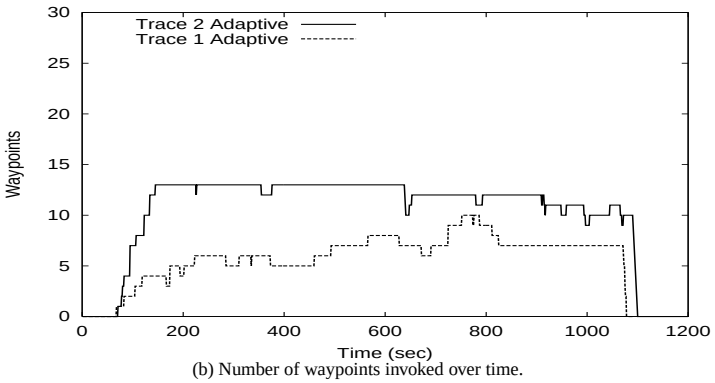
(c) *Environment2*, 10 min mean stay, 2 sec interarrival.

Fig. 3 Importance of Adaptive invocation: Performance in environments with different dynamicity.

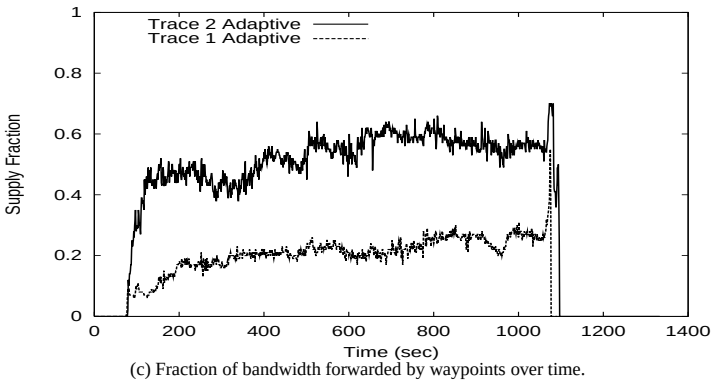
and *Static-RI(1.5)* in the more dynamic environment. To get a sense of the number of waypoints invoked by the scheme in each environment, consider Figure 3(a). This shows the mean *ORI* for both environments using the adaptive invocation algorithm. The *NRI* is also shown for comparison purposes, and note that this is the same for both environments. The mean *ORI* is 1.25 in *Environment1*, and 1.4 in *Environment2*. The *ORI* values match the the



(a) Average bandwidth received by participants with and without waypoints.



(b) Number of waypoints invoked over time.



(c) Fraction of bandwidth forwarded by waypoints over time.

Fig. 4 Effectiveness of waypoints in low NRI environments.

Static-RI thresholds that provided good performance for the two environments, indicating the the *Adaptive* scheme is invoking the appropriate number of waypoints in each environment.

Overall the results indicate the potential of waypoints to improve performance by enhancing system stability. Furthermore, the desired *ORI* is dependent on the environment,

and the *Adaptive* scheme has potential to automatically set and reach an appropriate *ORI* that can provide good performance while invoking only the needed waypoints.

5.2 Effectiveness in low NRI environments

In this section, we evaluate the effectiveness of the waypoint invocation scheme in enhancing performance in bandwidth constrained environments. Our evaluations are conducted on Planetlab with two different trace segments, each 1000 seconds in duration, and with member characteristics summarized in Table 1. We observe that *Trace2* is more resource-constrained, with a larger fraction of members behind NATs and firewalls, and a larger fraction of members that are unable to stream the source rate. The mean *NRI* is about 0.8 for *Trace1*, and 0.6 for *Trace2*. A set of 19 Planetlab hosts are used as waypoints, and a source rate of 400Kbps is used for the broadcasts.

Figure 4(a) depict the average bandwidth received by each participant in *Trace1* and *Trace 2* with and without the use of waypoints. The two curves toward the right show the performance without waypoints. The performance is poor for both traces, with 25% and 35% of the receivers seeing less than 90% of the source rate. The two curves toward the top-left of Figure 4(a) depict the performance when using the adaptive algorithm to invoke and remove waypoints. Over 93% of the members receive greater than 90% of the bandwidth in both traces, indicating the effectiveness of the invoked waypoints in achieving good performance. While there is a tail, and the performance of a few incarnations is not improved, further analysis revealed that these incarnations were being emulated on a particular Planetlab node that suffered from congestion. Invoking waypoints cannot help improve performance since the poor performance is due to congestion local to receivers.

We next consider the number of waypoints invoked to achieve good performance. In both traces, the mean *ORI* achieved was about 1.1, which indicates only necessary waypoints were invoked. Figure 4(b) shows the number of waypoints that are part of the broadcast over time. The adaptive invocation algorithm maintained between 5 and 7 waypoints in the group during most of *Trace1* and between 10 and 13 waypoints in *Trace2*. The algorithm correctly invoked more waypoints in *Trace2*. Figure 4(c) shows the percentage of bandwidth forwarded by waypoints over time for both traces. About 20% of the bandwidth was forwarded by waypoints in *Trace1*. In *Trace2* waypoints forwarded 50% to 60% of the data. We believe this is reasonable given the *NRI* was roughly 0.8 and 0.6 for the two trace segments.

5.3 Responsiveness of Invocation Schemes

In this section, we evaluate the responsiveness of our system, i.e. how long it takes for a waypoint to improve the performance of members in the system. To understand this further, we consider the time taken from when the waypoint is invoked to when each stage of the waypoint invocation process is completed. These include the time taken for (i) the waypoint to begin the process of connecting to the data delivery structure (*Waypoint Join Time*); (ii) the waypoint to connect to the data delivery structure, and start receiving data (*Connect Time*); (iii) the waypoint to receive the full stream bandwidth and start accepting children (*Eligible Parent Time*); (iv) the first child to connect to the waypoint (*Child Connect Time*); and (v) the child to receive the full stream bandwidth from the parent (*Waypoint Utilization Time*).

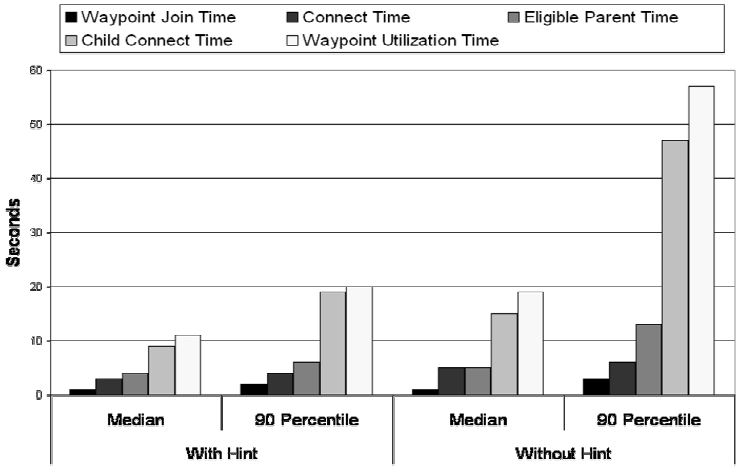


Fig. 5 Time taken for each stage of waypoint invocation to be completed. All time values are measured from when a waypoint is invoked, and include the time taken for previous stages.

Figure 5 shows the time values measured from our experiments. We consider the median and $90th$ percentile of these times across all waypoint invocations. The first two sets of bars in Figure 5 represent the median and $90th$ percentile for our system. We see that all time values are low - the *Waypoint Utilization Time* is within 20 seconds for 90% of the invocations.

As discussed in Section 3.3, our system employs a heuristic (*WaypointHint*) to improve system responsiveness where a waypoint is provided with a list of poor performing members to ensure a waypoint is effectively utilized as soon as it is introduced. To illustrate the performance improvement provided by this heuristic consider the last two bars in Figure 5 which represent the timing when the system does not use this heuristic. The *Waypoint Utilization Time* has a median of 19 seconds, and a 90th percentile of 57 seconds. The largest contributor to the long delay is the time from when a waypoint is eligible to take children to the time when a child connects to the waypoint and receives the full streaming bandwidth. As demonstrated, providing the hint significantly reduces this time, resulting in an overall reduction of a factor of 2 in the median *Waypoint Utilization Time*, and a factor of 3 in the 90th percentile of the *Waypoint Utilization Time.*

5.4 Effectiveness of Waypoint Selection

In this section, we explore the importance and effectiveness of our heuristics for intelligent waypoint selection in ensuring good system performance. To stress our system, we add 13 Planetlab hosts to the waypoint set that are known to have poor bandwidth performance to other Planetlab hosts [30]. We refer to the original and new set of waypoints as the *Waypoints-BaseSet*, and the *Waypoints-ExtendedSet* respectively.

We evaluate two versions of our system with the *Waypoint-ExtendedSet*: *WaypointRating* and *NoWaypointRating*. The *WaypointRating* scheme rates the performance of waypoints and removes waypoints that do not provide good performance to their children as discussed in Section 3.3. The *NoWaypointRating* scheme does not have this feature enabled.

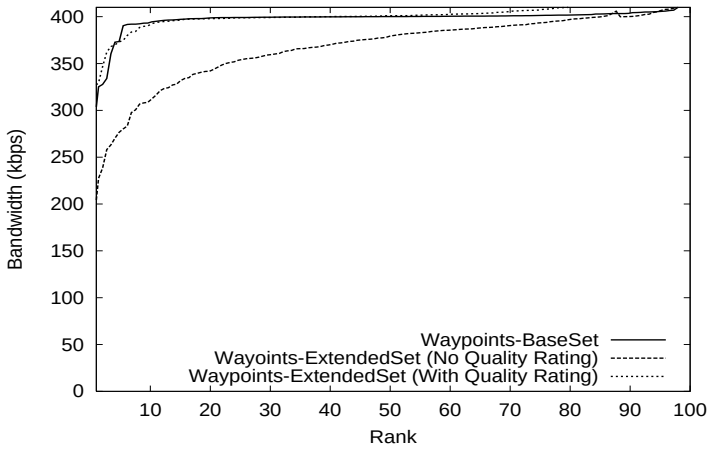


Fig. 6 Effectiveness of intelligent waypoint selection:Average bandwidth received by each member.

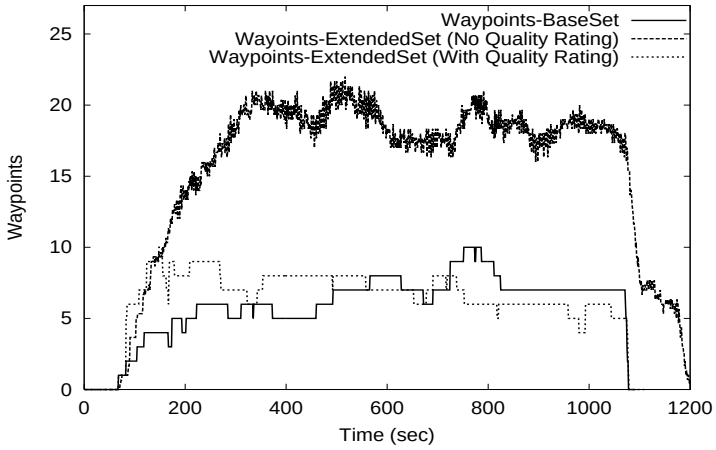


Fig. 7 Effectiveness of intelligent waypoint selection: Number of waypoints invoked over time.

Figure 6 shows the average bandwidth received by participants using the *Waypoint-ExtendedSet*, with and without the rating scheme. For comparison purposes, performance with the *Waypoint-BaseSet* scheme is also shown. The performance with rating is good and almost indistinguishable from the performance with the *Waypoint-BaseSet*, while the performance is significantly worse without the rating scheme as depicted by the lowest line in Figure 6.

Figure 7 compares the schemes with respect to the number of waypoints invoked. When the rating scheme is used with the *Waypoint-ExtendedSet*, less than 10 waypoints are invoked, which is comparable to the number invoked with the *Waypoint-BaseSet*. However, without the rating scheme, more than 20 waypoints were invoked.

Our findings show that invoking waypoints without considering their quality can hurt system performance, and highlight the importance and the effectiveness of the *WaypointRating* scheme. The benefits with *WaypointRating* accrue because the waypoint invoker has ac-

cess to the collective performance data from all of the waypoint's children and can quickly determine that a waypoint is poor.

6 Conclusion

The performance achievable with live P2P video broadcast systems is limited by the intrinsic characteristics of participating members such as user dynamics, bandwidth constraints and connectivity constraints (NATs). In this paper, we have shown how performance-driven, and on-demand invocation of waypoints, can overcome these limitations. The on-demand invocation, as well as the fact that waypoints themselves consume bandwidth resources and could hurt the system if not carefully selected, distinguish our approach from hybrid P2P/CDN architectures, and from seeds in BitTorrent.

We have presented a system design to realize the waypoint architecture. The key elements include (i) an adaptive waypoint invocation algorithm that considers both node performance and environment resources, and dynamically estimates the appropriate number of waypoints to invoke; (ii) mechanisms to rate waypoint quality, and intelligently select only those waypoints that can enable good performance; and (iii) self-monitoring mechanisms where participating members can themselves monitor the performance and availability of bandwidth resources in the system.

Through detailed evaluations on Planetlab and Emulab, we have shown the system ensures good performance while only invoking an appropriate number of waypoints in a range of realistic environments with different dynamicity and *NRI* values. Our results show the importance and effectiveness of our adaptive waypoint invocation scheme. For instance, while an *ORI* of 1.2 is sufficient to achieve good performance in *Environment1*, a higher *ORI* is required in the more dynamic *Environment2*, making a scheme based on achieving static *ORI* thresholds insufficient. In contrast, our scheme is able to adaptively pick appropriate *ORI* values for both environments. Our results also highlight the importance and effectiveness of our heuristics for intelligent waypoint selection. In experiments with the *Waypoint-ExtendedSet*, over 93% of members see more than 90% of the source rate when *WaypointRating* is enabled, while only less than 70% of members see such performance when the heuristic is disabled.

While our evaluations have been conducted using tree-based protocols, we believe the waypoint architecture can also benefit mesh-based protocols, and much of the system design is independent of the underlying protocol. While we do not explicitly consider incentive mechanisms for waypoints in this paper, we believe that a large body of ongoing research on incentive mechanisms in overlay multicast [1, 4, 21, 23, 24, 29, 32, 34, 37] can be easily integrated with the waypoint architecture. That said, customizing the waypoint architecture to mesh-based designs, and integrating existing incentive mechanisms are interesting directions for future research.

References

1. A.Habib and J. Chuang. Incentive mechanism for peer-to-peer media streaming. In *Proceedings of the 12th IEEE International Workshop on Quality of Service (IWQoS'04)*, 2004.
2. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, Aug. 2002.
3. S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller. Construction of and Efficient Overlay Multicast Infrastructure for Real -time Applications. In *Proceedings of IEEE INFOCOM*, Mar. 2003.

4. M. Bishop, S.G.Rao, and K. Sripanidkulchai. ”Considering Priority in Protocols for Overlay Multicast”. In *Proceedings of IEEE Infocom*, 2006.
5. S. Buchegger and J. Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Proceedings of the Second Workshop on Economics of Peer-to-Peer Systems*, June 2003.
6. M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Content Distribution in Cooperative Environments. In *Proceedings of SOSP*, 2003.
7. Y. 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 *Proceedings of USENIX*, June 2004.
8. Y. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM Sigmetrics*, June 2000.
9. B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of First Workshop on the Economics of Peer-to-Peer Systems*, 2003.
10. D. Dutta, A. Goel, R. Govindan, and H. Zhang. The design of a distributed rating scheme for peer-to-peer systems. In *Proceedings of the First Workshop on Economics of Peer-to-Peer Systems*, June 2003.
11. P. Francis. Yoid: Your Own Internet Distribution, http://www.aciri.org/yoid/. April 2000.
12. A. Ganjam and H. Zhang. Connectivity restrictions in overlay multicast. In *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Aug. 2004.
13. Geobytes, Inc. Database for Mapping IP Addresses to Geographic Coordinates. http://www.geobytes.com/.
14. I. Gupta, R. van Renesse, and K. P. Birman. A probabilistically correct leader election protocol for large groups. In *International Symposium on Distributed Computing*, pages 89–103, 2000.
15. C. Huang, J. Li, A. Wang, and K. Ross. Understanding hybrid cdn-p2p: Why limelight needs its own red swoosh. In *Proceedings of ACM NOSSDAV*, May 2008.
16. Y. Huang, T. Z. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p vod system. In *Proceedings of ACM SIGCOMM*, August 2008.
17. J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, Oct. 2000.
18. T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution. In *Proceedings of ACM IMC*, 2005.
19. D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of SOSP*, 2003.
20. H. T. Kung and C.-H. Wu. Differentiated admission for peer-to-peer systems: Incentivizing peers to contribute their resources. In *Proceedings of the First Workshop on Economics of Peer-to-Peer Systems*, June 2003.
21. Z. Liu, Y. Shen, K. Ross, S. Panwar, and Y. Wang. Substream trading: Towards an open p2p live streaming system. In *Proceedings of IEEE ICNP*, October 2008.
22. N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *Proceedings of IEEE INFOCOM*, May 2007.
23. J. D. Mol, D. H. P. Epema, and H. J. Sips. The orchard algorithm: Building multicast trees for p2p video multicasting without free-riding. *IEEE Transactions on Multimedia, vol. 9, no. 8*, December 2007.
24. T. Ng, D. Wallach, and P. Druschel. Incentives-compatible peer-to-peer multicast. In *Proceedings of the Second Workshop on Economics of Peer-to-Peer Systems*, June 2004.
25. T. E. Ng and H.Zhang. Predicting internet network distance with coordinates-based approaches. In *In Proceedings of IEEE INFOCOM*, June 2002.
26. V. Padmanabhan, H. Wang, and P. Chou. Resilient Peer-to-peer Streaming. In *Proceedings of IEEE ICNP*, Nov. 2003.
27. V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proc. The 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2005.
28. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *Proceedings of 3rd Usenix Symposium on Internet Technologies & Systems (USITS)*, March 2001.
29. F. Pianese, D. Perino, J. Keller, and E. W. Biersack. Pulse: an adaptive, incentive-based, unstructured p2p live streaming system. *IEEE Transactions on Multimedia, vol. 9, no. 8*, December 2007.
30. PlanetLab. PlanetLab IPerf. http://www.planet-lab.org/logs/iperf/.
31. SETI@Home. http://setiathome.ssl.berkeley.edu/.
32. V. Shrivastava and S. Banerjee. Natural selection in p2p streaming: From the cathedral to the bazaar. In *Proceedings of ACM NOSSDAV, Skamania, WA*, June 2005.
33. K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of peer-to-peer architectures for large-scale live streaming application. In *Proceedings of the ACM SIGCOMM*, Aug. 2004.

34. Y. Sung, M. Bishop, and S. Rao. Enabling contribution awareness in an overlay broadcasting system. In *Proceedings of ACM SIGCOMM*, 2006.
35. B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *OSDI02*, pages 255–270, Boston, MA, Dec. 2002.
36. D. Xu, H. K. Chai, C. Rosenberg, and S. Kulkami. Analysis of a hybrid architecture for cost-effective streaming media distribution. In *ACM MMCN*, 2003.
37. S. Yuen and B.Li. Strategyproof mechanisms for dynamic multicast tree formation in overlay networks. In *Proceedings of IEEE Infocom*, April 2005.
38. X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. Donet/coolstreaming: A data-driven overlay network for live media streaming. In *In Proceedings of IEEE INFOCOM*, Mar. 2005.