Multisource Video On-Demand Streaming in Wireless Mesh Networks

Yong Ding, Yang Yang, and Li Xiao, *Senior Member, IEEE*

***Abstract*—We study the multisource video on-demand (*VoD*) application in multichannel multiradio wireless mesh networks. When a user initiates a new video request, the application can stream the video not only from the media servers, but also from the peers that have buffered the video. The multipath multisource video on-demand streaming has been applied in wired networks with great success. However, it remains a challenging task in wireless networks due to wireless interference. In this paper, we first focus on the problem of finding the maximum number of high-quality and independent paths from the user to the servers or peers for each *VoD* request by considering the effect of wireless interference. We formulate it as a constrained maximum independent paths problem and propose two efficient heuristic path discovery algorithms. Based on the multiple paths discovered, we further propose a joint routing and rate allocation algorithm, which minimizes the network congestion caused by the new *VoD* session. The algorithm is aware of the optimization for both existing and potential *VoD* sessions in the wireless mesh network. We evaluate our algorithms with real video traces. Simulation results demonstrate that our algorithm not only improves the average video streaming performance over all the coexisting *VoD* sessions in the network, but also increases the network's capacity of satisfying more subsequent *VoD* requests.**

***Index Terms*—Multipath routing, multisource video streaming, rate allocation, wireless mesh network.**

I. Introduction

THE VIDEO-ON-DEMAND (*VoD*) application has become a popular Internet service recently. There have already been several commercial products developed to support *VoD* applications, such as PPLive and PPStream. Most of them use peer-to-peer (P2P) technology to improve the *VoD* performance. Such architecture has been discussed in [1]. Assume users can store the videos that they have recently watched in their local storage (e.g., PPLive and PPStream buffer 1 GB of the most recently watched videos, which is enough for over two 2-h movies, in a peer's local storage). When a client wants to watch a new video, he or she first discovers which peer clients have buffered the video, and then streams the video from both the servers and peer clients through multiple paths. The multipath multisource video on-demand streaming has been applied in wired networks with great success. However, it remains a challenging task in wireless networks due to the effect of wireless interference.

As wireless mesh networking technology is attracting more interest in research and industry, it is envisioned to be used for low-cost infrastructures for last-mile Internet access and building community networks [2]. A community network is a static multihop wireless network composed of many mesh routers, where each mesh router establishes connectivity with neighboring mesh routers. There are some special routers working as gateways to provide access to the Internet. In a large community network, when a *VoD* user initiates a video request, it can stream the video from two types of sources: 1) the servers or peers that have buffered the video within the community network; 2) even if there are no such sources, the user can stream the video from multiple servers or peers in the Internet through the multiple gateways. As *VoD* applications have high demand of bit rate, delay, and loss sensitivity, the bottleneck of the multipath video streaming is usually in the community network, that is, the multihop wireless paths from the user to peers or servers in the community network or the paths from the user to gateways. Fortunately, in multichannel multiradio wireless mesh networks, the enhanced channel diversity increases the network capacity. In this paper, we study multipath routing and rate allocation in multichannel multiradio wireless mesh networks to improve the *VoD* performance.

One major problem for the multisource *VoD* application in wireless mesh networks is the discovery of multiple independent paths. By splitting the video stream over multiple independent paths, we can not only improve the aggregate routing performance, but also improve the stability and robustness. In the Internet, the independent paths are usually defined as edge-disjoint or vertex-disjoint paths. In edge-disjoint paths, no two paths share a same link, and therefore any link failure will only affect one path. Vertex disjointedness is stronger than edge disjointedness because it also guarantees that any node failure will affect at most one path. In wireless mesh networks, the discovery of independent paths becomes more challenging due to wireless interference. Even if two paths are edge-disjoint or vertex-disjoint, they might still affect each other if they have wireless links that interfere with each other. Thus, their aggregate routing performance becomes lower than expected, and the congestion of one path will probably influence the other path. This route coupling effect has been studied in [3]. Therefore, in order to find independent paths in wireless mesh networks, we should guarantee interference-disjointedness in addition to edge-disjointedness or vertex-disjointedness.

Another challenge is that the optimization of *VoD* performance should be considered over all *VoD* users in the network instead of only the current user. As we know, the wireless mesh

Manuscript received March 10, 2011; revised August 25, 2011 and January 21, 2012; accepted January 21, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Reisslein.

The authors are with the Department of Computer Science and Engineering, Michigan State University, MI 48824 USA (e-mail: dingyong@cse.msu.edu; yangyan5@cse.msu.edu; lxiao@cse.msu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2188642

network is designed to be shared among multiple users. If the routes with required bandwidth have been found for a *VoD* request, the *VoD* user can establish connections in the network for data transport, which we call a *VoD* session. There might be multiple coexisting *VoD* sessions from different users in the wireless mesh network. Thus, we should not be too selfish when finding the routes and rate allocation for the current *VoD* request. Instead, we should consider not only the resulting performance of the current *VoD* session, but also its influence on the other existing *VoD* sessions in the network and the network's ability of satisfying new *VoD* requests.

In this paper, we consider the wireless interference in the discovery of multiple independent paths. For each arriving *VoD* request, given n senders (servers or peers), which can provide the same video, we find the maximum number of $m(m \leq n)$ high-quality and independent paths that connect from m senders to the receiver (the new user who initiates the *VoD* request). We formulate it as a constrained maximum independent paths problem and design efficient path discovery algorithms. Based on the multiple paths discovered, we further propose a joint multipath routing and rate allocation algorithm to determine the routes together with the rate allocated for each route for the *VoD* request, with the goal of minimizing the network congestion. By using this optimization framework, we not only improve the average performance of existing *VoD* sessions, but also enable the network to support more subsequent *VoD* requests. We evaluate our algorithms by using video trace simulations. The rest of this paper is organized as follows. In Section II, we summarize the previous work. In Section III, we present the system model. In Section IV, we describe two proposed heuristic algorithms for multipath discovery. In Section V, we propose a joint routing and rate allocation algorithm. The simulation methodology and results are shown in Sections VI and VII, and we conclude our work in Section VIII.

II. Related Work

Video streaming is a challenging task due to its high bit rate, delay, and loss sensitivity. It is believed that the application performance can be improved by streaming the video over multiple paths. In [1], [14], and [15], a peer-to-peer architecture has been proposed, where a user can stream videos from both servers and peers. There are two major problems with multisource video streaming: 1) multipath discovery, which finds multiple independent paths; 2) rate allocation, which determines the sending rate for each source of the path.

Multipath routing has been used in wireless ad hoc networks to provide error resilience and load distribution. A number of multipath routing protocols [6], [4], [5] have been proposed to discover multiple paths between a single source and a single destination. These protocols focus on finding edge-disjoint paths. However, there still exists correlation between these paths due to wireless interference.

There have been several studies of multipath video streaming in wireless ad hoc networks. In [16], the authors tried to find optimal paths for each session such that the overall video distortion is minimized. However, they have not considered wireless interference in the problem formulation. In [7], [8], and [17], the authors took the wireless interference into consideration, and studied the selection of optimal paths between a single source and a single destination. The studies optimized for different objectives. Reference [7] aims at minimizing the distortion metric [18], while [8] and [17] try to minimize concurrent packet drop probability over all paths. In contrast, our work focuses on multipath streaming from multiple senders (peers or servers) to one receiver.

Multipath video streaming over static wireless mesh networks has been studied in [9] and [10]. The authors used double-description coding and tried to find two paths to optimize the distortion of the video streaming application. However, the proposed mechanisms are for single-channel wireless mesh networks and do not guarantee the independency among the two paths. As a result, the failure or congestion of one path may affect the other. Moreover, their optimization only considers splitting the video stream over two paths. In contrast, our work tries to find multiple independent (interference-disjoint) paths in multichannel wireless mesh networks, which not only improves video streaming performance, but also increases its robustness. Our multipath routing and rate allocation algorithms consider a more general case, where the video stream is split over multiple paths (not limited to two paths) with appropriate rate on each path. In [19], the authors proposed a distributed channel assignment, routing, and rate allocation scheme for video streaming over wireless mesh networks. It assumes that each user streams from a single video source over multiple paths. In contrast, our work focuses on a more general P2P application, in which each user can stream video from multiple sources over multiple paths.

A similar problem has been studied in other networks. In [20], the authors studied multipath selection in Internet overlay networks with the goal of maximizing average video quality. In [21], the authors proposed to build a tree to aggregate videos from multiple video surveillance nodes (leaf nodes) to a center (root node) in sensor networks while minimizing the wireless interference in the tree. In contrast, our work focuses on multipath video streaming in multichannel multiradio wireless mesh networks.

In multichannel multiradio wireless mesh networks, the WCETT [22] is a commonly used metric to evaluate the quality of a single path. It jointly considers the delay, packet loss rate, and channel diversity in the selection of a high-quality single path. In our multipath discovery algorithms, we use this metric as the criterion when we try to find each single path. We use the algorithm proposed in [23] to find an optimal path based on the WCETT metric in this paper.

There have been many studies on rate allocation for a single video streaming session, assuming the multiple routes have already been found. They have proposed rate allocation schemes to minimize the packet loss [11], or to be used with *FEC* [12], or minimize the distortion metric [13]. All these rate allocation algorithms optimize for the current session only. In contrast, our rate allocation algorithm is aware of both the performance of existing *VoD* sessions and the network's capability of satisfying potential *VoD* requests.

There are also some studies on rate allocation schemes at higher level, that is, to determine the rate of each video

streaming session, assuming each video has been encoded into different rates [24], [25]. In this paper, we assume each video has a fixed rate (such as PPLive and PPStream). We focus on the rate allocation on the multiple paths for each *VoD* session, such that the sum of rates on the multiple paths satisfies the total rate requirement of the video.

III. Multisource VoD in WMN

In this section, we introduce the system model. We will propose our solutions in Sections V and VI. Some preliminary work has been introduced in [26].

Consider the *VoD* application in a large community network. The network is composed of a number of multichannel multiinterface wireless mesh routers. Each mesh router can establish wireless connectivity with neighboring mesh routers, so that a static multihop wireless network is formed. There are some special mesh routers working as gateways, which are directly connected to the Internet. Previous static channel allocation algorithms, such as [27] and [28], can be used to assign each interface of each router with a channel so as to minimize the network interference while maintaining the network connectivity. Compared to static channel allocation, dynamic channel allocation can achieve better adaptivity to traffic. However, it also incurs considerable overhead with channel switching, especially when each node has multiple interfaces. There have been several dynamic channel allocation algorithms proposed in [29] and [30]. A channel assignment and scheduling algorithm that considers channel switching overhead in optimization has been proposed in [31]. In this paper, we focus on the study of routing and rate allocation problem for multisource video streaming. We will leave the joint consideration of channel allocation with routing and rate allocation as future research direction.

For some popular videos, the *VoD* performance can be improved by P2P technology. Whenever a user has requested a video, it registers to the server, so that the server keeps a list of the users that have buffered the video. When a new user visits, he or she queries the server for the list of users, from which they can stream the video. If there are such users, the new user can stream the video not only from the media server, but also from the other peers. For the peers that are located in the community network, the user downloads over a multihop wireless path, while for the peers in the Internet, the user downloads over a path, which consists of a multihop wireless path from the user to one of the gateways and a path from the gateway to the peer in the Internet.

We call the peer or the server that provides the download of the video the *sending node*, and the user that requests the video the *receiving node*. A mesh router is called a *sender* (or *receiver*) if it is connected with one or more sending nodes (or the receiving node). If the sending node is located in the community network, the mesh router with which the user is directly connected is the sender. If the sending node is in the Internet, the gateway through which users in the community network can access it is the sender. In the example illustrated in Fig. 1, where there is a media server in the Internet, we assume $P1$, $P2$, A, C are the peers that have buffered the video. If X wants to watch the video, there are five available senders in this case, which include $R1$, $R2$, and the three gateways.

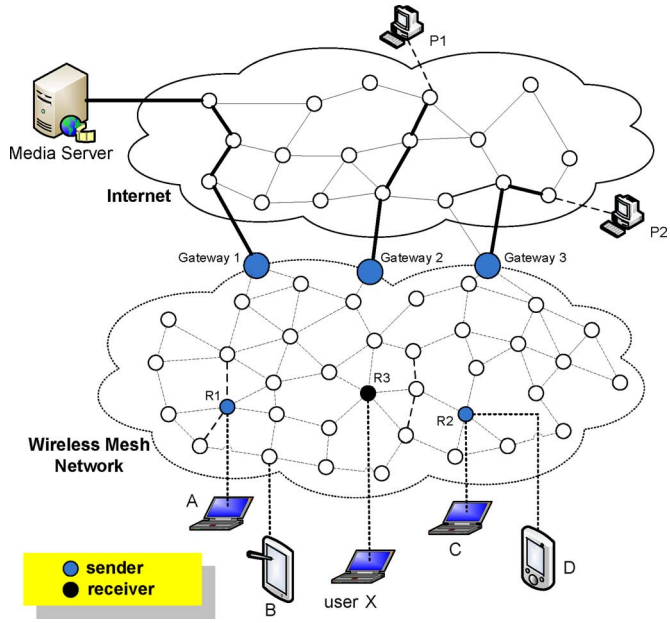


Fig. 1. VoD in wireless mesh networks.

For each arriving *VoD* request, we need to determine the routes from senders to the receiver and the rate on each route for video streaming. A *VoD* request can be satisfied if we find the routes with the required bandwidth that convey the demanded video quality; otherwise, the *VoD* request cannot be satisfied, and it will be blocked. If a *VoD* request can be satisfied, the receiver will establish connections for video streaming in the network. For a *VoD* request, from the time when the connections are established to the time when they are closed due to the end of video streaming, we call the set of connections a *VoD session*.

In the following, we will address two problems gradually: 1) the discovery of multiple paths (we will formulate the problem and propose some heuristic multipath discovery algorithms in Section IV-B); 2) how to allocate the rate on the paths (we will propose a joint routing and rate allocation algorithm in Section V).

IV. Multipath Discovery

A. Problem Formulation

Consider a wireless mesh network $G(V, E)$, where V denotes the set of mesh routers whose locations are known. There is an undirected edge $(u, v) \in E$ if and only if $d(u, v) \leq R$, where $d(u, v)$ is the Euclidean distance between mesh routers u and v, and R denotes the maximum wireless radio transmission range.

Suppose each mesh router is equipped with Q interfaces, and there are K orthogonal channels available. Given a channel assignment A, where $A(u)(u \in V)$ denotes the set of channels assigned to the interfaces of u, the *topology* $G_A(V, E_A)$ of the network can then be determined. There is a wireless link $e = (u, v, c) \in E_A$ if and only if $(u, v) \in E$ and $c \in A(u) \bigcap A(v)$, that is, u and v each have at least one interface working on channel c.

We use the protocol model to determine whether two wireless links in G_A interfere with each other or not. For any two

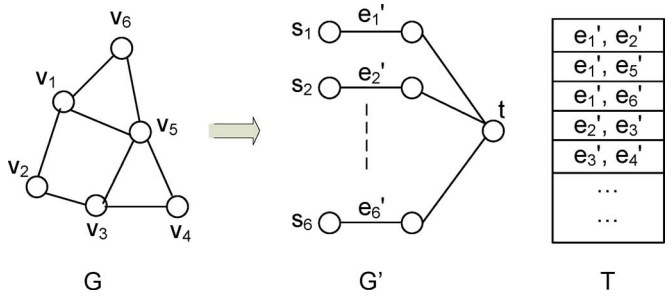


Fig. 2. Problem reduction.

links $e_i, e_j \in E_A$, define their distance $d(e_i, e_j)$ as the minimum Euclidean distance between any node of one link and any node of the other link. e_i and e_j interfere with each other if: 1) $c(e_i) = c(e_j)$, where $c(e)$ denotes the channel of wireless link e; 2) $d(e_i, e_j) \leq I$, where I is the wireless interference range, which is usually $2R$.

Definition 1: Given topology $G_A(V, E_A)$, the *conflicting table* is the set of all link pairs (e_i, e_j), where $e_i, e_j \in E_A$ and $e_i \neq e_j$, such that e_i and e_j interfere with each other.

In the topology $G_A(V, E_A)$, let $r \in V$ be the receiver, and assume there are n senders $S = \{s_1, s_2, \ldots, s_n\} \subseteq V$. We consider the case where a mesh router will not be a sender and a receiver at the same time (explained in Section III), that is, $r \notin S$.

Definition 2: Given topology G_A and its conflicting table T, two paths p and q interfere with each other if there exists $(e_i, e_j) \in T$ such that $e_i \in p$ and $e_j \in q$. The *Maximum Interference Disjoint Paths problem* (MIDP) seeks the maximum number of edge-disjoint paths from S to r, denoted by P, where each path is between a different sender in S and r, such that there does not exist (p_i, p_j), where $p_i, p_j \in P$ and $p_i p_j$ interfere with each other.

In other words, the problem finds the maximum number of paths from the set of senders to the receiver, such that any two paths are independent from each other with regard to wireless interference. Let $n = \mid S \mid$ be the number of senders and $m = \mid P \mid$ be the number of paths, hence we have $m \leq n$.

Theorem 1: The MIDP problem is NP-hard.

Proof: We can prove this by reducing the maximum independent set problem, which is NP-complete, to the formulated problem. Given an undirected graph $G(V, E)$, the maximum independent set problem finds the largest independent set (no two nodes from the set are adjacent) in the graph. We first transform the graph G to G' as follows. 1) For each $v_i \in V$, create an edge e'_i in G'. Denote one end vertex of e'_i by s_i, and create an edge from the other end vertex to a vertex t. Fig. 2 shows an example of the transformation. 2) Create a list T, which is initialized to empty. For each $e = (v_i, v_j) \in E$, add (e'_i, e'_j) to T. As a result, the maximum independent set problem can be reduced to the problem of finding maximum interference-disjoint paths from $S = \{s_1, s_2, \ldots, s_{|V|}\}$ to t in G', given the conflicting table T. The reduction only takes polynomial time. Therefore, the formulated problem is NP-hard. ■

In the real-world *VoD* applications, we not only want to find more interference-disjoint paths from S to r, but also need to guarantee the quality of each path with regard to packet loss, delay, and throughput. There have been many studies on metrics for finding good routes between a single source and a single destination in wireless networks. The WCETT metric [22] is widely used in multichannel multiinterface wireless mesh networks. It not only considers packet loss and delay, but also accounts for channel diversity in each path so as to reduce intraflow interference. Therefore, we use this metric to evaluate the quality of each selected path. More specifically, we want to find the maximum number of independent paths, while keeping the WCETT value of each path below a certain threshold. The WCETT metric of a path can be calculated as follows:

$$\mathrm{WCETT} = (1-k) * \sum_{i=1}^{n} \mathrm{ETT}_i + k * \max_{1 \leq j \leq c} X_j$$

where ETT_i is the expected transmission time of a packet on the link i, and X_j is the sum of transmission times of hops on channel j

$$X_j = \sum_{\text{Hop } i \text{ is on channel } j} \mathrm{ETT}_i.$$

Another problem in real applications is that there might not be enough completely interference-disjoint paths in some cases because of the limited number of channels and number of interfaces. To take advantage of multipath streaming, we can relax the constraint on the path independency a little bit to allow more paths to be found, which improves the robustness of applications.

Definition 3: Given a set of edge-disjoint paths P in topology G_A, consider any link $e \in p_0 (p_0 \in P)$. The set of paths that interfere with e is $I(e) = \{p \mid p \in (P - p_0) \bigwedge p$ has a link that interfere with $e\}$. The *path interference* of link e is defined as $\mathrm{PI}(e) = \mid I(e) \mid$, which reflects how many other paths in P are interfering with this link. The *maximum path interference* of P is defined as $\mathrm{MPI}(P) = \mathrm{Max}_{p \in P} \mathrm{Max}_{e \in p} \mathrm{PI}(e)$.

Now we can formally describe the problem by considering both the constraint and relaxation.

Definition 4: Given topology G_A and its conflicting table T, the *COnstrained Maximum INdependent Paths problem* (COMINP) seeks the maximum number of edge-disjoint paths from S to r, denoted by P, such that: 1) $\mathrm{MPI}(P) \leq \alpha$, where α is the threshold to control the level of independency between paths in P; 2) $\mathrm{WCETT}(p) \leq \beta$ for $p \in P$, where β is the threshold to control the quality of each path.

Note that by setting $\alpha = 0$ in the first constraint and $\beta = +\infty$ in the second constraint, the COMINP problem becomes exactly the MIDP problem. Therefore, COMINP is also NP-hard.

B. Multipath Discovery Algorithm

In this section, we propose two heuristic algorithms, the Iterative Path Discovery algorithm (IPD) and the Parallel Path Discovery algorithm (PPD). We list the definitions of some frequently used notations in Table I. Some notations will be explained in detail when we present the algorithms. Both algorithms run in a centralized fashion on the receiver.

In wireless mesh networks, the mesh routers usually have minimal mobility. For example, in community networks, the routers are usually fixed on roofs of houses. In addition, due

TABLE I
NOTATIONS

$T(V, E)$	the original network topology
$T'(V, E')$	the remaining network topology
S	the set of senders
r	the receiver
$IF_T(p)$	total interference of path p in topology T
$IE_T(p)$	path interfering set of p in topology T
$l(e)$	the number of paths interfering with e
$WCETT(p)$	the $WCETT$ value of path p

to the overhead of dynamic channel switching, static channel allocation strategies are widely used, in which the channel assignment does not change often. This makes it possible for each mesh router to collect the global knowledge of the network, including each other router's position and channel assignment. This can be done by letting each router broadcast the information to the whole network each time the network topology has been reconstructed or channels have been reassigned, which occurs very rarely. Therefore, each mesh router knows the global topology of the wireless mesh network (wireless links and channels), which makes it possible to use a centralized algorithm on the receiver to find multiple paths from senders to the receiver.

1) Iterative Path Discovery: The Iterative Path Discovery algorithm (IPD) finds paths one by one from the senders to the receiver. In each iteration, we find one path from a sender to the receiver, and then update the topology accordingly. This process continues until no new paths can be found from the remaining topology. There are two critical steps in the algorithm, which we explain in the following.

a) Path Selection: Let S' be the set of remaining senders, for which we have not found paths yet, and T' be the remaining topology. Initially, $S' = S$ and $T' = T$. In this step, for each $s \in S'$, we first find a minimum WCETT path p from s to r in T' if such a path exists and $\mathrm{WCETT}(p) \leq \gamma \cdot w_T(s, r)$, where $w_T(s, r)$ is the WCETT value of the optimal path from s to r in T and γ is a constant to control the quality of each path. (We set $\gamma = 1.5$ based on our experiments to achieve a good balance between the number of paths that could be found and the optimality of each path. This is because if we strictly require that each path selected must be the optimal WCETT path, there is less chance of finding multiple paths with satisfying the path independent constraint.) Denote the resulting set of paths as P. We then need to decide which path to select from P. As the algorithm aims at discovering as many paths as possible in the final solution, we select a path from P based on the following metric.

We define the *total interference of p in* $T'(V, E')$, denoted by $\mathrm{IF}_{T'}(p)$, as the number of edges in T' that interfere with any edge in p. More formally

$$\mathrm{IF}_{T'}(p) = \mid \{e' \mid e' \in E' \bigwedge \exists e \in p : \mathrm{Interfere}\ (e, e')\} \mid .$$

Note that $\mathrm{Interfere}(e, e')$ is true if $e = e'$.

Therefore, we select the path from P, which has the minimum total interference in T'. The reason is that the selected path will render minimum change in the remaining topology, and thus leave us more flexibility in finding more paths afterwards.

Algorithm 1: Iterative Path Discovery Algorithm

1: $T'(V, E') = T(V, E)$, $S' = S$
2: $l(e) = 0$ for $e \in E'$, $X = \{\}$
3: **repeat**
4: For each $s_i \in S'$, find a minimum WCETT path p_i from s_i to r in T' if $\mathrm{WCETT}(p) \leq \gamma \cdot w_T(s, r)$. Denote the set of paths by P.
5: Select p^* from P, which has the minimum total interference in T', $\mathrm{IF}_{T'}(p^*) = \mathrm{Min}_{p_i \in P}\mathrm{IF}_{T'}(p_i)$.
6: $l(e) = l(e) + 1$ for $e \in \mathrm{IE}_{T'}(p^*)$
7: $T' = T' - p^*$
If $l(e) > \alpha$, $T' = T' - e$, for $e \in \mathrm{IE}_{T'}(p^*)$
8: $S' = S' - \mathrm{sender}(p^*)$, $X = \{X, p^*\}$, where $\mathrm{sender}(p^*)$ is the sender of path p^*
9: **until** P is empty
10: X is the set of selected paths.

b) Topology Update: Once a path p has been selected from T', we need to update T' accordingly. We can guarantee the edge disjointedness of paths by taking off p from T', so that the paths found later will not overlap with the paths previously found. In addition, we need to consider p's interference on T' and guarantee the level of independency among the final set of selected paths.

We use a label l on the edges of T' to record the interference from the already selected paths, where $l(e)$ counts how many selected paths are interfering with link e. At the start of the algorithm, $l(e)$ is initialized to 0. Assume p is the selected path from $T'(V, E')$ in the current iteration. We define the *path interfering set of p in* T', denoted by $\mathrm{IE}_{T'}(p)$, as the set of edges in $(T' - p)$ that interfere with any edge in p.

$$\mathrm{IE}_{T'}(p) = \{e' \mid e' \in (E' - p) \bigwedge \exists e \in p : \mathrm{Interfere}(e, e')\}.$$

We update $l(e)$ for each edge $e \in \mathrm{IE}_{T'}(p)$ by increasing 1, indicating that there is one more selected path p that interferes with e. If $l(e) > \alpha$, then we need to take off e from T'. This is because if e has been used in one more path in the remaining topology, then e will interfere with more than α already selected paths, which violates the constraint in the COMINP problem.

The algorithm is formally described in Algorithm 1. In each iteration, we select a path and update the topology, which takes $O(\mid S \mid\mid E \mid\mid V \mid)$. There are at most $\mid S \mid$ iterations, so the algorithm takes polynomial running time $O(\mid S \mid^2 \mid E \mid\mid V \mid)$.

2) Parallel Path Discovery: IPD finds a complete path in each step, which limits the search space, and thus may not be able to approximate the optimal solution. Instead of searching for paths one by one, we take a different approach in this section, that is, we perform a parallel search for paths from all senders to the receiver level by level. The basic idea of the Parallel Path Discovery algorithm (PPD) is as follows. In each iteration, we perform a breadth-first search from r in the remaining topology $T'(V, E')$ and categorize all the nodes in V into layers based on their distances from r. We start from the senders that are in the farthest layer and find partial paths that connect as many of them as possible to nodes in the lower

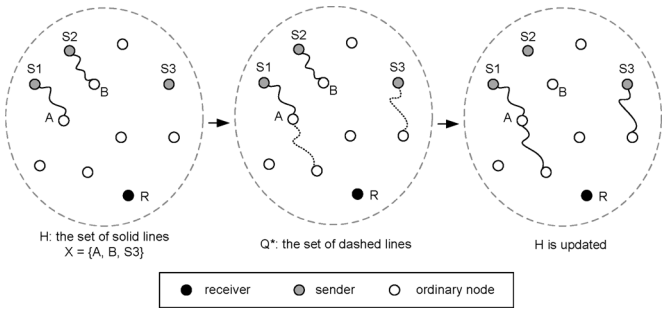


Fig. 3. Basic idea of parallel path discovery.

layer. We then use these nodes that are reached in the lower layer as new senders to take place of the original senders. The process continues until we reach the receiver. The critical step of the algorithm is finding partial paths in each layer, so that we can guarantee that the paths finally found satisfy the constraints. In the following, we first illustrate the overall algorithm in Section IV-B.2a, and then describe how to find partial paths in each layer in Section IV-B.2b.

a) Parallel Search: As illustrated in Fig. 3, let $H = \{p_1, p_2, \ldots, p_l\}$ be the set of partial paths found in the current iteration. Each partial path $p_i \in H$ starts from a sender in S, denoted by $s(p_i)$ and ends at a nonreceiver node, denoted by $e(p_i)$. We do a breadth-first search on the current remaining topology T'. Let $d(v)(v \in V)$ be the distance from v to the receiver r in T'. Consider the set of nodes $X = \{e(p_i)\} + (S - \{s(p_i)\})$, which contains the set of ending nodes of the partial paths and the set of senders that do not have partial paths yet. These are the nodes from which we can further extend the set of partial paths.

In each iteration, we extend partial paths only from the highest-layer nodes in X, that is, $X' = \{v \mid d(v) = d_{\max}, v \in X\}$ where $d_{\max} = \mathrm{Max}_{v \in X} d(v)$. We want to find as many paths as possible that connect the nodes from X' to the next lower layer nodes $Y = \{v \mid d(v) = d_{\max} - 1\}$. Assume Q^* is the set of paths found to extend H. Let $H_{Q^*} \subseteq H$ be the set of partial paths that can be extended by paths in Q^*. We extend these partial paths and make them as the new H. In addition, we also need to update T'. We take off paths in Q^* together with their interfering edges whose l values exceed the threshold from T', and also recovering paths in $(H - H_{Q^*})$ back into T'. After we have updated H and T', we do a breadth-first search again on the updated topology and continue to extend H following the same procedure.

The algorithm is described in Algorithm 2. There is a key subalgorithm *LayerPathSearch* used to extend the set of partial paths to the next lower layer, which we will explain in the following.

b) Layer Path Search: We first enumerate all the paths of at most 3 hops long from X' to Y in T'. The reason for doing this is the following. 1) If we only use one hop path to extend the partial paths, we are always finding the shortest paths. However, in multichannel wireless mesh networks, the channel diversity is also important for path quality in addition to hop distance. Therefore, we also check two or three hop paths in order to utilize the channel diversity in the path selection. 2) If we do not limit the path length, we will have exponential search space. In practice, we find that by keeping the threshold at 3, we have enough flexibility in finding channel diverse paths.

Algorithm 2: Parallel Path Discovery Algorithm

1: $T' = T$, $H = \{\}$
2: **repeat**
3: Do a breadth-first search on T'
4: $Q^* = \mathrm{LayerPathSearch}(H, T')$
5: Extend H and update T' based on Q^*
6: **until** H reaches the receiver r
7: Output all the paths in H.

Algorithm 3: FindPathSet(H, T')

1: $Q^* = \{\}$
2: In T', find each path q that is at most 3 hops from X' to Y such that $\mathrm{WCETT}(p+q) \leq \gamma \cdot w_T(\mathrm{sender}(p), r)$, where $p \in H$ concatenates with q. Denote the set by Q.
3: Rate each path $q \in Q$ by metric $z(q)$.
4: **repeat**
5: Take the path q^* with the lowest z value in Q.
6: $Q^* = Q^* + q^*$
7: Update T' and Q based on q^*.
8: **until** no paths can be selected
9: Q^* is the set of paths to extend H

Denote the set of paths by Q; we want to find as many paths as possible from Q to extend H without breaking the edge disjointedness and path independency constraints. We rate each path $q \in Q$ based on two factors.

1) The interference caused by q on T', denoted by $\mathrm{IF}_{T'}(q)$. The path that causes less interference is preferred to be selected because it increases the possibility of finding other more paths later.
2) The interference caused by q on the connecting partial path $p \in H$, that is, q starts from the ending node of p. We denote it by $\mathrm{IF}_p(q)$. This is related with the channel diversity of each final path discovered. While we want to find more paths, we also need to guarantee the quality of each path. Therefore, q can be evaluated by $z(q) = k_1 \times \mathrm{IF}_{T'}(q) + k_2 \times \mathrm{IF}_p(q)$.

The algorithm is described in Algorithm 3. We take the path with the lowest z value from Q each time. When we have selected a path $q \in Q$ to extend a partial path $p \in H$, we need to update T' in the same way as described in the iterative path discovery algorithm. We must not only take off q from T', but also take off edges whose l value exceeds the threshold α in T'. As some edges have been taken off from T', we also need to update Q because some paths may become unavailable in the updated topology. We continue to select paths from Q until Q becomes empty. As a result, we get the set of paths to extend H.

Given H in the current iteration, it is possible that H contains too many partial paths so that they are occupying too many link resources in the lower layers due to wireless interference (these links have been removed from the original topology because

Algorithm 4: LayerPathSearch(H, T')

1: $H' = H$
2: $Q^* = \mathrm{FindPathSet}(H', T')$
3: **repeat**
4: For each $p_i \in H'$, release p_i and update T', $Q_i = \mathrm{FindPathSet}(H' - p_i, T')$
5: Let Q_k be the best solution among $\{Q_i\}$
6: **if** Q_k is better than Q^* **then**
7: $Q^* = Q_k$, $H' = H' - p_k$
8: **end if**
9: **until** no better solution can be found
10: Q^* is the path set to extend H.

their l value exceeds the threshold). This may dramatically reduce the number of paths finally discovered. To deal with this problem, we can release some partial paths from H first, so that we can have enough link resources in the lower layers, and thus we may be able to extend more partial paths to the next layer. The algorithm is described in Algorithm 4, which can be summarized as trying to find a local minimum point.

Let D be the diameter of T. In Algorithm 3, we consider $O\left(\mid S \mid \left(\frac{\mid V \mid}{D}\right)^3\right)$ partial paths, and thus the running time is $O\left(\mid S \mid \left(\frac{\mid V \mid}{D}\right)^3 \mid E \mid\right)$. Algorithm 4 calls Algorithm 3 $O(\mid S \mid^2)$ times, and thus takes $O\left(\mid S \mid^3 \left(\frac{\mid V \mid}{D}\right)^3 \mid E \mid\right)$. There are $O(\mid D \mid)$ iterations in Algorithm 2. Therefore, the algorithm takes polynomial running time $O\left(\frac{\mid S \mid^3 \mid V \mid^3 \mid E \mid}{\mid D \mid^2}\right)$.

Although the parallel path discovery algorithm takes more computation overhead than the iterative path discovery algorithm, it has the promise of finding more paths. In the iterative path discovery algorithm, we find one complete path each time and modify the remaining topology based on the path. As a result, we have made a big change on the remaining topology, and soon no new paths could be found. In contrast, the parallel path discovery algorithm finds partial paths in each step and leaves more flexibility in the remaining topology for finding more paths.

C. Discussion

If we want to find strictly edge-disjoint paths, the number of paths that can be found will be limited by the number of interfaces that the receiver has. In order to find more paths to enhance the robustness of the application and provide more flexibility for rate allocation, we can relax this constraint by allowing paths to merge at the last hop toward the receiver. Both the iterative and parallel path discovery algorithms can be easily modified to deal with this case. When taking off a selected path from the remaining topology, we do not remove the edge that is directly connected with the receiver. It will be taken off from the remaining topology only when its l value is over the threshold α. In this way, we have still guaranteed the path independency constraint on the last hop.

V. Joint Routing and Rate Allocation

As the wireless mesh network is designed to be shared among multiple users, the routing and rate allocation for each *VoD* request should not only consider the performance of itself, but also take into account the existing *VoD* sessions and the network's ability of satisfying more subsequent *VoD* requests. Note that the performance of each existing *VoD* session is dramatically affected by the traffic load on each link used by the session. If the traffic load on a link is high, the application may experience high queuing delay and jitter due to the congestion on this link. In addition, the increase in the number of congested links may disrupt the network's connectivity, thereby causing the network to be incapable of finding routes with required bandwidth for new *VoD* requests. Therefore, we take our optimization goal as minimizing the network congestion.

In this section, we first propose an optimal rate allocation algorithm on the multiple discovered paths, which determines the rate on each path. It is possible that some discovered paths may be unused (or allocated with zero rate) because they are more congested than others. We then propose a joint routing and rate allocation algorithm with the goal of minimizing the network congestion. Unlike previous works [11]–[13] that optimize for the performance of a single session only, our optimization not only improves the performance of existing sessions, but also improves the network's capability of satisfying more subsequent *VoD* requests. The algorithm runs in a centralized fashion on the receiver. We assume each router periodically broadcasts the available bandwidth of its neighboring wireless links to all the routers in the network, so that each router knows the available bandwidth on all the links in the network (similar to [27]).

Assume the multiple paths discovered for a *VoD* request is $P = \{p_1, p_2, \ldots, p_m\}$, and the receiver needs to determine the optimal data rate on each path. Let r_k be the data rate on path $p_k (k = 1, 2, \ldots, m)$, and R be the total data rate required by the *VoD* session. We have

$$\sum_{k=1,\ldots,m} r_k = R. \tag{1}$$

The traffic load on each link $e_i \in P$, denoted by $t(e_i)$, is

$$t(e_i) = \sum_{p_k \in P \bigwedge e_i \in p_k} r_k. \tag{2}$$

Let $A(e_i)$ be the available bandwidth on link e_i. Let IE_{ij} denote whether the two links e_i and e_j interfere with each other. $\mathrm{IE}_{ij} = 1$ if it is true, and $\mathrm{IE}_{ij} = 0$ if otherwise. The residue capacity of each link $e_i \in P$ under the rate allocation $\{r_1, r_2, \ldots, r_m\}$, denoted by $C(e_i)$, is

$$Z(e_i) = A(e_i) - t(e_i) - \sum_{e_j \in P \bigwedge e_j \neq e_i} t(e_j) \times \mathrm{IE}_{ij}. \tag{3}$$

Therefore, our optimization goal is to maximize the residue capacity of the bottleneck link (or minimize the network congestion)

$$\text{Maximize } \mathrm{Min}_{e_i \in P}\{Z(e_i)\} \tag{4}$$

Algorithm 5: Joint Routing and Rate Allocation

1: Let C be the capacity of each link, $A(e)$ be the available bandwidth of link e. Set $\mathrm{Thresh} = C$.
2: Generate subtopology $T[\mathrm{Thresh}]$, which only includes links e where $A(e) \geq C - \mathrm{Thresh}$.
3: Find paths P for the session on $T[\mathrm{Thresh}]$. Calculate optimal rate allocation r on P.
4: Set $P^* = P$, $r^* = r$. Let v^* be the value of the objective function under (P, r) ((4)).
5: **repeat**
6: $\mathrm{Thresh} = \mathrm{Thresh}/2$.
7: Find paths P for the session on $T[\mathrm{Thresh}]$. Calculate optimal rate allocation r on P.
8: Let v be the value of the objective function.
9: **if** $v \geq v^*$ **then**
10: $P^* = P$, $r^* = r$, $v^* = v$
11: **end if**
12: **until** (no valid P or r exists $\|v < v^*$)
13: return (P^*, r^*)

The problem (1)–(4) is a max-min linear programming problem, and can be transformed to a general linear programming (LP) problem. Therefore, we can solve for the optimal rate allocation $\{r_1, r_2, \ldots, r_m\}$ in polynomial time. Note that it is not mandatory for all the paths in P to be used for the *VoD* request. It is possible for some paths to be allocated with rate of zero by solving (4) because they are more congested than others. In other words, *the multipath discovery algorithm gives candidate paths, while the rate allocation algorithm [by solving (4)] determines the subset of paths to be used together with the rate on each path for the VoD request*.

To approximate the optimal joint routing and rate allocation, we use binary search in Algorithm 5. We use IPD or PPD for path discovery subalgorithms. They not only find multiple high-quality and independent paths, but also are better at balancing the traffic in the network. By finding edge-disjoint paths, the traffic can be well distributed over the network spatially. By minimizing the interference among paths, the traffic can be well distributed over different channels. In Algorithm 5, we first apply path discovery (IPD or PPD) and rate allocation (LP) algorithms sequentially on the original topology to find an initial solution. We then truncate the topology based on a threshold in each step, that is, we only keep the links with enough available bandwidth. We apply the same subalgorithms and find new solutions. The search terminates if we cannot find any better solutions by decreasing the threshold.

After a receiver has calculated the routing and rate allocation for a *VoD* request, it needs to reserve the bandwidth on the paths. It is possible that the reservation fails because the bandwidth has already been reserved by some other *VoD* request arriving at almost the same time. More specifically, when the receiver calculates the routing and rate allocation, its knowledge of the current network status does not get updated of the other *VoD* session in time. In this case, we can resolve it by letting the receiver wait for a random time and recalculate the rate allocation on the multiple paths already discovered until it succeeds with reserving the bandwidth. For the application in a real wireless mesh network, this case happens rarely because it takes a short time for a receiver to calculate the routing and rate allocation for a *VoD* request (less than 0.3 s) and to reserve bandwidth together with updating link status (less than 0.4 s) in a 60-nodes wireless mesh network (see Section VII-G). According to the statistics of all the *VoD* requests for a popular *VoD* server over the Internet [32], the *VoD* request arrival rate is around one request per second. As a result, we can expect that the mean interarrival time of *VoD* requests in a regional wireless mesh network should be at least in minutes. For Poisson process (with the mean interarrival time of 1 min), the probability that two or more requests arrive within 0.7 s is less than 1.2%, which is a very small probability.

In the *VoD* application, after a receiver has established a *VoD* session, some senders might become unavailable due to the following reasons: 1) when the peer continues to watch new movies and its buffer is full, the oldest movie will be removed from the buffer; 2) there is a link failure or node failure in the network, which makes the path to some sender disconnected. An intuitive way is to run the algorithm on the network again. However, this will affect the paths that the receiver is currently using. A better way is to use the algorithm to find more paths in the rest of the topology without affecting the existing paths used. However, if the event of any path failure or any sender leaving the network happens more than a certain number of times on the session, it is better to rerun the algorithm for it.

VI. Simulation Methodology

In this section, we introduce the tools, traces, methods, and metrics that we use in the simulation.

A. Methodology

We perform the simulations in *NS2*. The Hyacinth extension [33] has been used to support multiple channels and multiple interfaces per node in the simulator. In all the simulations, we set the maximum radio transmission range to 250 m and the interference range to 500 m. We use 802.11 with bit rate of 11 Mb/s for each interface and use UDP with the maximum packet size of 1024 B for multimedia data transport. Unless specifically stated, the simulation is performed in a 60-nodes random topology within an area of 1500×1500 m^2. Each mesh router is equipped with four interfaces, and there are eight orthogonal channels used in the channel assignment. We use the channel allocation algorithms proposed in [28] to statically assign channels to each interface in order to minimize the wireless interference within the network.

We use the video traces available in the public domain [34]–[37]. Each video trace file includes the size of each encoded video frame together with its quality. During *NS2* simulation, the frames are encapsulated into UDP packets during network transmission and reconstructed at the receiver. Table II illustrates the three video traces we use for evaluation. Each movie is 30 min long. All the traces use group of pictures (GoP) length of 16 with I-frame as the first frame in each GoP and have a rate of 30 frames per second. All the trace files

TABLE II
VIDEO TRACES USED IN SIMULATION

Movie Name	Encoding	Quantization Scale	Average Bit Rate (bps)
Silence of the lambs	H.264, Single-Layer	(22, 22, 24)	358365.5
Star Wars IV	H.264, Single-Layer	(22, 22, 24)	373880.4
Die Hard	H.264, Single-Layer	(22, 22, 24)	369614.9

being used contain the frames in the encoder order. In other words, the frames are transmitted through the network in the encoder order.

We compare the following methods of path discovery for multisource video streaming.

1) MinW: Find a minimum WCETT path between each sender and the receiver. Thus, if the network is connected and there are n senders, this method will find n paths.
2) MEDP: Find the maximum number of edge-disjoint paths from the senders to the receiver (edge-disjoint paths have been used in both the Internet [20] and multihop wireless networks [4]).
3) IPD: Use the iterative path discovery algorithm to find the maximum number of independent paths.
4) PPD: Use the parallel path discovery algorithm, which has the potential to find more paths than IPD under the same constraints. To be consistent in the comparison, we use the joint routing and rate allocation framework (Algorithm 5) proposed in Section V for all the methods. In other words, we use Algorithm 5 with different path discovery algorithms as subalgorithms to find multiple paths, while using the same rate allocation algorithm for all the methods.

B. Metrics

We evaluate different methods based on the following metrics in our simulation.

- The maximum number of concurrent *VoD* sessions that can be supported in the network: This metric reflects the network capacity of supporting *VoD* applications.
- Packet delivery delay: This metric determines the response time of the *VoD* application. As we are streaming data over multiple paths, each path may possess a different average packet delay. In this case, we calculate the maximum average packet delay over all paths.
- Packet delay jitter: This metric is critical for the quality of video streaming. For a single path, we can evaluate delay jitter by the variance of delays. Let $d_i (i = 1, 2, \ldots, k)$ be the delay of each packet in a single path within an interval of time, the delay jitter of $d = \{d_i \mid i = 1, 2, \ldots, k\}$ can be evaluated by $\mathrm{jitter}(d) = \sqrt{\sum_i (d_i - \bar{d})^2 / k}$, where $\bar{d} = \sum_i d_i / k$. We can extend this metric to multiple paths in the following way. Let $d_{ij} (i = 1, 2, \ldots, m; j = 1, 2, \ldots, k_i)$ be the delay of the jth packet in the ith path. The delay jitter of $d = \{d_{ij} \mid i = 1, 2, \ldots, m; j = 1, 2, \ldots, k_i\}$ can be evaluated by $\mathrm{jitter}(d) = \sqrt{\sum_i \sum_j (d_{ij} - \bar{d}_i)^2 / \sum_i k_i}$, where $\bar{d}_i = \sum_j d_{ij} / k_i$.
- Packet drop ratio: When a receiver streams a video over multiple paths, it needs to determine a playout deadline, that is, the time the receiver waits for before playing out the video. As a result, the deadline of each packet to be received can be determined. A packet drop occurs when: 1) the packet is lost due to the collision in wireless transmission; 2) the packet has been successfully received, but it is received after its deadline. The packet drop ratio is defined as the number of packets dropped over the total number of packets transmitted. It influences the quality of the decoded video.
- Percentage of frames reconstructed: Note that there are dependencies between the encoded video frames. For example, the I-frame in a GoP is required to decode all other P-frames and B-frames in the GoP, and the P-frame is required to decode all the successive P-frames as well as the B-frames encoded with respect to these P-frames. Thus, even if a frame has been received before deadline, it is regarded as *not constructed* if its dependent frames are not reconstructed successfully.
- Peak signal-to-noise ratio (PSNR): This is a commonly used metric to evaluate the quality of a decoded video. In the video trace file, the PSNR value is computed for each frame based on the difference between the decoded frame and the original frame.
- Processing and session setup overhead: This includes the time to compute the multiple paths together with rate allocation and the time needed to establish the connections for each session.

Note that "packet delivery delay," "packet delay jitter," and "packet drop ratio" are network-layer metrics, which give us some insight on the perceived video quality, while "percentage of frames reconstructed" and "PSNR" are application-layer metrics, which directly assess the perceived video quality. In this paper, we will evaluate both categories of metrics.

VII. PERFORMANCE EVALUATION

In this section, we present the simulation results from *NS2* based on real video traces.

A. Number of Paths

We select one router in the network as the receiver and randomly designate n routers in the network as senders. MinW always finds n paths, while the number of paths discovered by MEDP is $\min\{n, \mathrm{degree}(r)\}$, that is, the network is well-connected so that the number of edge-disjoint paths to the receiver r is bounded by its degree in the topology G_A. For both IPD and PPD, we set the threshold $\alpha = 0, 1$ (see α in Definition 4). By setting $\alpha = 0$, we require that each path does not interfere with any other path. When $\alpha = 1$, each link of any path interferes with at most one other path among the multiple paths finally discovered. According to Section IV-C, we allow paths to merge at the last hop towards the receiver for both IPD and PPD.

Unlike MinW and MEDP, IPD and PPD take wireless interference into consideration when finding multiple paths, and thus have the prospect of providing better video streaming quality. The number of paths discovered by IPD and PPD under different number of senders is illustrated in Fig. 4. Each point corresponds to the average of 20 runs. We can observe that PPD is able to discover more paths than IPD under the same constraint. As a result, PPD provides more flexibility for finding

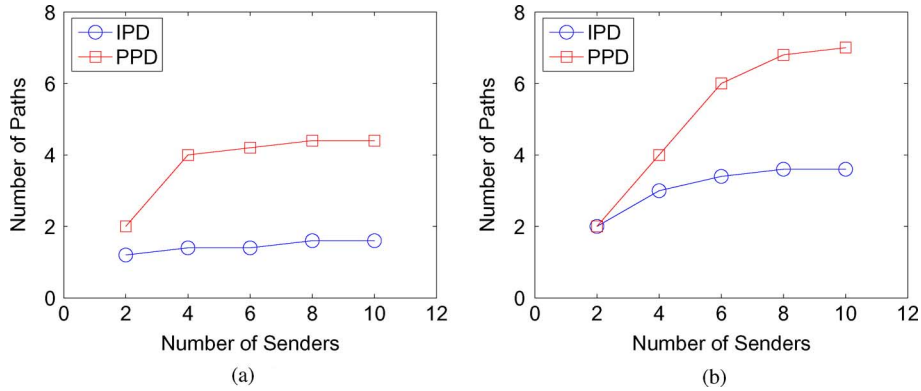


Fig. 4. Number of paths discovered. (a) $\alpha = 0$. (b) $\alpha = 1$.

appropriate rate allocation to minimize the network congestion than IPD. Especially when $\alpha = 0$, IPD finds only a single path in most cases, while PPD discovers more paths so that we can use multipath streaming to improve the performance of video streaming. Both algorithms tend to find more paths with the increase of available senders.

B. Number of Sessions

We experiment on networks of different scales (with the same density as the default 60-node topology) without changing the other default settings. In each network, three mesh routers are randomly selected as initial senders. They may be mesh routers connected with a peer that has buffered the video, or a gateway through which a local user can access a peer or media server in the Internet. Assume there are three popular movies recently (listed in Table II), and each of the initial senders can provide all of the three movies. Assume *VoD* requests arrive in accordance with Poisson distribution. For each arriving *VoD* request, the user is connected to a random mesh router in the network and initiates a *VoD* request for a movie that is randomly selected from the three movies. We use different algorithms to find the routing and rate allocation for each arriving *VoD* request. If the *VoD* request can be satisfied and has been established successfully, it becomes a sender, which can provide the movie to subsequent *VoD* requests. This process continues until there is a *VoD* request that cannot be satisfied. In this way, we can get the maximum number of concurrent sessions that can be supported in the network. Based on our experiments, it does not make a visible difference in the maximum number of concurrent sessions with the order of the *VoD* request. This is because it is quite flexible to allocate resources for each session if we use multiple paths. Hence, in the following evaluation, we do not make difference between the order of arriving requests.

Fig. 5 demonstrates the maximum number of concurrent sessions supported by each method under different network scales. Each point has a confidence interval of $[\mathrm{AVE} - 1, \mathrm{AVE} + 1]$ (AVE is the average value at each point) with a confidence level of 0.95. For IPD and PPD, α is set to 1. From the figure, we can observe that MinW performs the worst in supporting multiple sessions. MEDP is slightly better than MinW. IPD supports over 10% more sessions than MEDP on average, while PPD improves the capacity by over 25% compared to MEDP. This is because IPD and PPD not only find edge-disjoint paths,

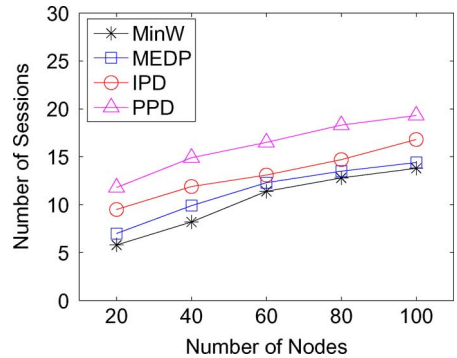


Fig. 5. Maximum number of sessions versus network scale (eight channels).

but also minimize the interference among the paths. Therefore, the traffic of each session can be well balanced over the network both spatially and over different channels. PPD excels IPD because PPD finds more paths than IPD under the same constraint, and thus PPD provides more flexibility for rate allocation to minimize network congestion than IPD. Note that the rate allocation algorithm does not mandate every discovered path for a single session to be allocated with some positive rate (explained in Section V). Although PPD finds more paths than IPD, it does not necessarily mean that PPD uses more paths for actual video streaming than IPD. The rate allocation on PPD paths usually has lower congestion than on IPD paths. Therefore, PPD supports more concurrent *VoD* sessions than IPD.

We repeat our simulations with different numbers of channels. Fig. 6 illustrates the maximum number of concurrent sessions under different numbers of orthogonal channels for different methods. We can observe that more concurrent *VoD* sessions can be supported in the network with the increasing number of channels because the network capacity is increased by using more orthogonal channels.

Note that the maximum number of sessions is not big compared to the 11-Mb/s bandwidth. This is because there is MAC protocol overhead and the interference is not completely eliminated in the network.

C. Delay and Jitter

We use the same assumptions of the video sources and the arrival of *VoD* requests as in Section VII-B. The video frames are encapsulated into UDP packets with the maximum size of

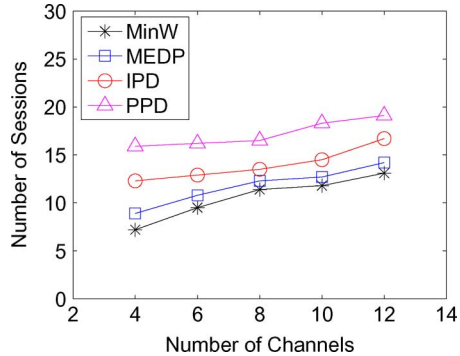


Fig. 6. Maximum number of sessions versus number of channels (60 nodes).

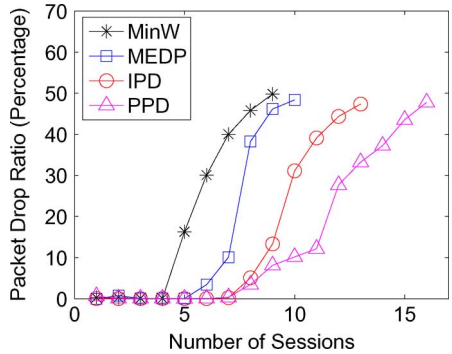


Fig. 9. Packet drop ratio over multiple *VoD* sessions (playout deadline = 300 ms).

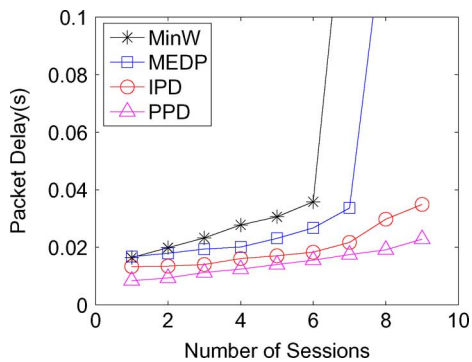


Fig. 7. Packet delay over multiple *VoD* sessions.

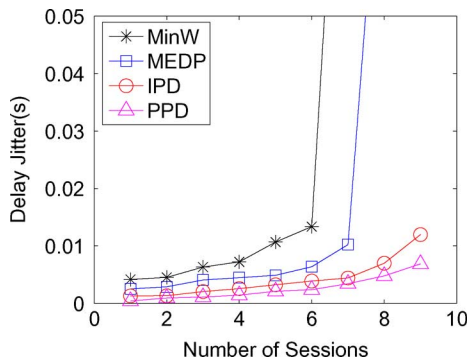


Fig. 8. Delay jitter over multiple *VoD* sessions.

1024 B for transmission (similar to [13]). In this scenario, the maximum number of concurrent sessions that can be supported by MinW, MEDP, IPD, and PPD is 9, 10, 13, and 16, respectively. If the network reaches its maximum capacity, additional *VoD* requests will be blocked. We calculate the average packet delay and delay jitter under different numbers of concurrent sessions in the network. The results are shown in Figs. 7 and 8.

In both figures, the horizonal axis shows the number of concurrent *VoD* sessions, and the vertical axis shows the average delay and jitter over all the sessions. As we can see, IPD and PPD enjoy lower delay and jitter than MinW and MEDP. For example, when there are five concurrent sessions, IPD and PPD reduce the packet delay by 28% and 39%, and reduce the delay jitter by 34% and 57%, compared to MEDP. This is because IPD and PPD not only find multiple high-quality paths with minimized interference, but also are better at balancing the traffic in the network both spatially and over different channels, and thereby reducing the network congestion.

Note that our goal is to minimize the network congestion so that the average performance of both the current session and the existing session can be optimized. As we can see in Fig. 7, with the increase of number of concurrent sessions, the average performance of the sessions will be degraded for all algorithms because there is overall more traffic in the network. For example, if we currently have four sessions, the allocation for the fifth session will cause the average performance of all the five sessions to degrade a little compared to before. However, IPD and PPD will be better than other algorithms.

D. Packet Drop Ratio

In this section, we first set the playout deadline at the receiver to 300 ms and calculate the packet drop ratio of *VoD* sessions. We then vary the playout deadline to evaluate its impact on the video streaming quality. If the receiver streams a video from a gateway, we add a random delay conforming to normal distribution $N(100\text{ms}, 20\text{ms}^2)$ (the parameters are derived from the packet delays extracted from an experiment where we repeatedly send ping packets from a gateway to a video server) on the path to simulate the delay from the peer in the Internet to the gateway.

Fig. 9 illustrates the average packet drop ratio over all sessions when there are different numbers of concurrent *VoD* sessions in the network. We can observe that the drop ratio increases with the increasing number of sessions because the network is getting more and more congested. When there are few (less than five) concurrent session in the network, all methods enjoy very low packet drop ratio. However, when more sessions have been established, IPD and PPD have dramatically lower packet drop ratio than MinW and MEDP because IPD and PPD find better paths for streaming the video. For example, to guarantee an average packet drop ratio less than 20%, the maximum number of concurrent sessions that can be supported by MinW, MEDP, IPD, and PPD is 5, 7, 9, and 11, respectively.

Fig. 10 demonstrates the impact of playout deadline on the packet drop ratio. We simulate the case when there are eight concurrent *VoD* sessions in the network. The vertical axis shows the average packet drop ratio over all the eight sessions. We can observe that the drop ratio decreases with longer playout deadline for all the methods because the playback deadline is relaxed

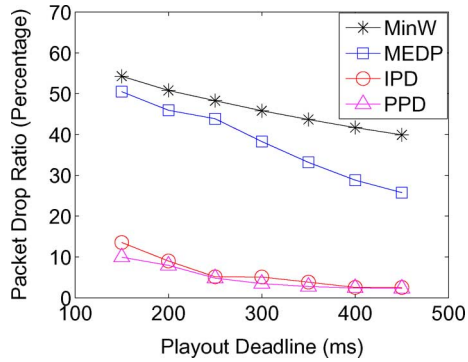


Fig. 10. Packet drop ratio under different values of playout deadline (eight sessions).

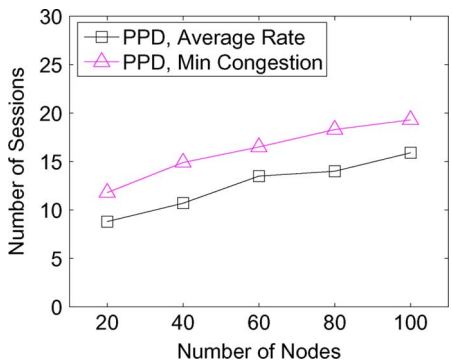


Fig. 11. Maximum number of sessions versus network scale (eight channels).

with longer playout deadline. For example, the packet drop ratio of IPD (or PPD) decreases from 14% (or 10%) to 3% (or 2%) when the playout deadline increases from 150 to 450 ms. In addition, IPD and PPD leads to much lower packet drop ratio than MinW and MEDP, which implies high video quality perceived at receivers. For example, at the playout deadline of 300 ms, IPD and PPD reduce the packet drop ratio by around 85%, compared to MEDP. We have also calculated the standard deviation of packet drop ratio over the multiple sessions for IPD and PPD, which is within 10% of the average value. This indicates that our algorithm does not have bias over a specific session in the case of multiple sessions.

E. Minimum Congestion Rate Allocation

In this section, we analyze how the rate allocation that minimizes the network congestion in Section V can help improve video streaming performance. We simulate Algorithm 5 with PPD for path discovery. We compared two different rate allocation algorithms: 1) average rate allocation on all the paths; 2) our proposed rate allocation algorithm that minimizes the network congestion. In Fig. 11, our proposed rate allocation substantially improves the maximum number of concurrent sessions that can be supported in the network. This is because by minimizing the network congestion, the network has more flexibility of satisfying subsequent sessions. Fig. 12 illustrates that by minimizing the network congestion, our proposed rate allocation also reduces the packet drop rate compared to the average rate allocation under the same number of concurrent sessions.

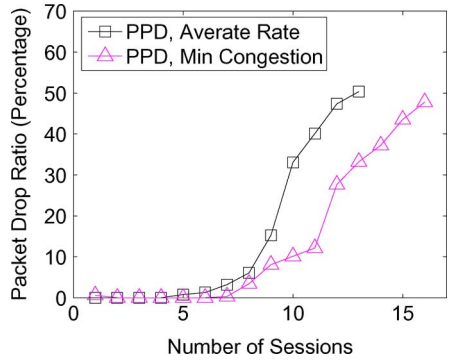


Fig. 12. Packet drop ratio over multiple *VoD* sessions (playout deadline $=$ 300 ms).

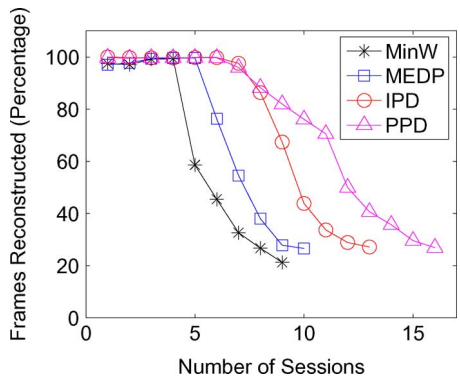


Fig. 13. Percentage of frames reconstructed over multiple *VoD* sessions (playout deadline $=$ 300 ms).

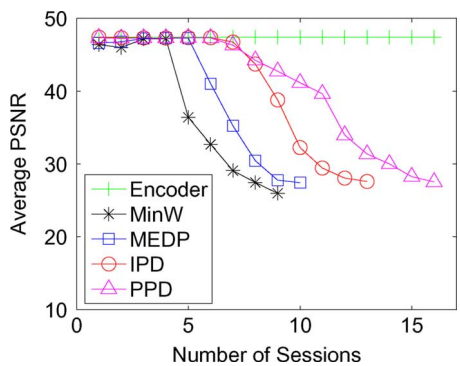


Fig. 14. Average PSNR over multiple *VoD* sessions (playout deadline $=$ 300 ms).

F. Perceived Video Quality

In addition to the network layer metrics that have been analyzed in the previous sections, we evaluate the application layer metrics of video streaming in this section.

Fig. 13 illustrates the percentage of frames reconstructed under each method when there are different numbers of concurrent *VoD* sessions in the network. Each point in the figure corresponds to the average of frame reconstruction ratio over all the sessions. The PSNR metric of videos perceived at receivers is calculated and demonstrated in Fig. 14. In the figure, each point corresponds to the average of PSNR over all the sessions, and has a confidence interval of $[\mathrm{AVE}(1-0.07),\ \mathrm{AVE}(1+0.07)]$ (AVE is the average value at each point) with a confidence level of 0.95. The "+" line plots the PSNR when there is no frame loss in video streaming (optimal value). The PSNR of the three

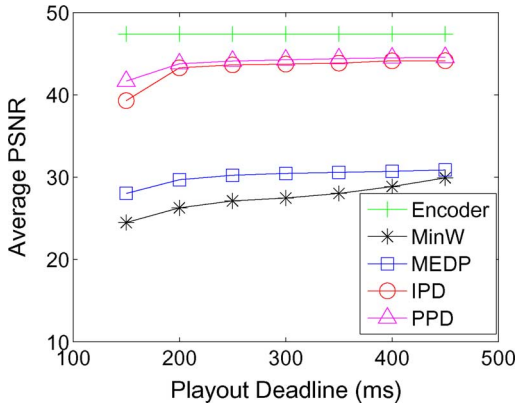


Fig. 15. Average PSNR under different values of playout deadline (eight sessions).

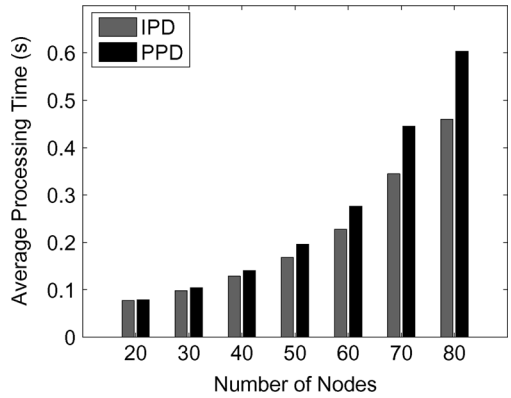


Fig. 16. Processing time for each *VoD* request.

movies in Table II without any frame loss is 47.7, 46.9, and 47.6 dB, respectively.

From both figures, we can observe that when there are fewer sessions in the network (less than five), all the algorithms lead to very high frame reconstruction ratio and high PSNR because of low packet drop ratio (shown in Fig. 9). However, when more sessions are established in the network, IPD and PPD obviously excel the other methods, while PPD performs better than IPD. For example, to guarantee an average PSNR no less than 40 dB, the maximum number of concurrent sessions that can be supported by MinW, MEDP, IPD, and PPD is 4, 6, 8, and 11, respectively.

Fig. 15 illustrates the PSNR under different values of playout deadline when there are eight concurrent *VoD* sessions in the network. We can observe that the perceived video quality increases with longer playout deadline. For IPD and PPD, when the playout deadline is over 200 ms, the improvement of PSNR becomes less obvious. In comparison, with the increase of playout deadline, there is steady improvement of PSNR for MinW and MEDP. This is because IPD and PPD have lower delay jitter than MinW and MEDP in network transmission. Therefore, IPD and PPD only need a small playout deadline to reach near-optimal performance.

G. Processing and Session Setup Overhead

We first evaluate the time needed for computing the routing and rate allocation for each *VoD* request. The processing overhead includes the discovery of multiple independent paths and the calculation of routing and rate allocation using LP. We apply our algorithms under different network scales. In each scenario, we set the number of senders to 8. We run the algorithms on a laptop computer with 2 GHz CPU. The average processing time is shown in Fig. 16. We can observe that PPD has a little longer processing time than IPD. However, PPD brings better performance than IPD with regard to the average session quality and the network's capacity based on the analysis in previous sections. Under the network scale of 60 nodes, both PPD and IPD are fast enough (less than 300 ms) to calculate the routing and rate allocation for each *VoD* request.

We also calculate the time needed for setting up the session once the routing and rate allocation have been determined for a *VoD* request. This includes reserving the bandwidth on the paths and letting the routers along the paths update the link status to the other routers. The simulation results in a 60-nodes network show that the session setup delay is less than 400 ms. Therefore, the total delay for processing and session setup is less than 700 ms for each *VoD* request.

VIII. Conclusion

The objective of this paper is to improve the performance of multisource *VoD* applications in multichannel multiradio wireless mesh networks. We first proposed two heuristic multipath discovery algorithms, IPD and PPD, to find multiple independent paths from senders to the receiver for each *VoD* request. The proposed algorithms consider wireless interference in the multipath discovery, so it is able to balance the video streaming traffic both spatially and on different channels in the network. Based on the multipath discovery algorithms, we then proposed a joint routing and rate allocation algorithm to find the routes and rate allocation with the goal of minimizing the network congestion. The algorithm not only optimizes the performance of existing *VoD* sessions in the network, but also improves the network's capability of satisfying more subsequent *VoD* requests. We performed simulations in*NS2* using real video traces and evaluated the performance of our algorithms using both network layer metrics and application layer metrics for video streaming. As part of our future work, we will analyze these algorithms in real systems. Simulation results have shown that IPD and PPD not only increase the maximum number of concurrent *VoD* sessions that can be supported in the network, but also improve the video streaming quality of each session compared to previous work. Moreover, PPD achieves better video streaming performance than IPD because it is able to discover more paths than IPD under the same constraint, and therefore provides more flexibility in finding rate allocation with minimized congestion. However, IPD runs faster than PPD, so it is more preferable in case when algorithm processing time is more important.

References

[1] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 1, pp. 121–133, Jan. 2004.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Comput. Netw.*, vol. 47, no. 4, pp. 445–487, 2005.

[3] M. R. Pearlman, Z. J. Haas, P. Sholander, and S. S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," in *Proc. ACM MobiHoc*, 2000, pp. 3–10.

[4] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *Proc. IEEE ICC*, 2001, vol. 10, pp. 3201–3205.

[5] M. Marina and S. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Proc. IEEE ICNP*, 2001, pp. 14–23.

[6] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Proc. Mobile Comput.*, 1996, pp. 153–181.

[7] S. Mao, Y. T. Hou, X. Cheng, H. D. Sherali, and S. F. Midkiff, "Multipath routing for multiple description video in wireless ad hoc networks," in *Proc. IEEE INFOCOM*, 2005, vol. 1, pp. 740–750.

[8] W. Wei and A. Zakhor, "Path selection for multi-path streaming in wireless ad hoc networks," in *Proc. IEEE ICIP*, 2006, pp. 3045–3048.

[9] D. Li, Q. Zhang, C.-N. Chuah, and S. J. B. Yoo, "Multi-source multipath video streaming over wireless mesh networks," in *Proc. IEEE ISCAS*, 2006, pp. 698–701.

[10] S. Kompella, S. Mao, Y. T. Hou, and H. D. Sherali, "Cross-layer optimized multipath routing for video communications in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 831–840, May 2007.

[11] T. P. Nguyen and A. Zakhor, "Distributed video streaming over internet," in *Proc. SPIE, Multimedia Comput. Netw.*, 2002.

[12] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Proc. Packet Video Workshop*, 2002.

[13] X. Zhu, S. Han, and B. Girod, "Congestion-aware rate allocation for multipath video streaming over ad hoc wireless networks," in *Proc. IEEE ICIP*, 2004, vol. 4, pp. 2547–2550.

[14] T. Nguyen and A. Zakhor, "Multiple sender distributed video streaming," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 315–326, Apr. 2004.

[15] Y. Zhu, W. Zeng, H. Liu, Y. Guo, and S. Mathur, "Supporting video streaming services in infrastructure wireless mesh networks: Architecture and protocols," in *Proc. IEEE ICC*, 2008, pp. 1850–1855.

[16] S. Mao, S. Kompella, Y. Hou, H. Sherali, and S. Midkiff, "Routing for multiple concurrent video sessions in wireless ad hoc networks," in *Proc. IEEE ICC*, 2005, vol. 2, pp. 1229–1235.

[17] W. Wei and A. Zakhor, "Interference aware multipath selection for video streaming in wireless ad hoc networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 165–178, Feb. 2009.

[18] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2006.

[19] L. Zhou, X. Wang, W. Tu, G.-M. Muntean, and B. Geller, "Distributed scheduling scheme for video streaming over multi-channel multi-radio multi-hop wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 3, pp. 409–419, Apr. 2010.

[20] A. Begen, Y. Altunbasak, and O. Ergun, "Multi-path selection for multiple description encoded video streaming," in *Proc. IEEE ICC*, 2003, vol. 3, pp. 1583–1589.

[21] V. Navda, A. Kashyap, S. Ganguly, and R. Izmailov, "Real-time video stream aggregation in wireless mesh network," in *Proc. IEEE PIMRC*, 2006, pp. 1–7.

[22] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio multi-hop wireless mesh networks," in *Proc. ACM MobiCom*, 2004, pp. 114–128.

[23] Y. Yang, J. Wang, and R. Kravets, "Interference-aware loop-free routing for mesh networks," UIUC, Urbana, IL, Tech. Rep., 2006.

[24] X. Zhu, P. Agrawal, J. P. Singh, T. Alpcan, and B. Girod, "Rate allocation for multi-user video streaming over heterogenous access networks," in *Proc. ACM Multimedia*, 2007, pp. 37–46.

[25] X. Zhu, P. Agrawal, J. P. Singh, T. Alpcan, and B. Girod, "Distributed rate allocation policies for multi-homed video streaming over heterogeneous access networks," *IEEE Trans. Multimedia*, vol. 11, no. 4, pp. 752–764, Jun. 2009.

[26] Y. Ding, Y. Yang, and L. Xiao, "Multi-path routing and rate allocation for multi-source video on-demand streaming in wireless mesh networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2051–2059.

[27] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and QoS routing in multi-channel wireless mesh networks," in *Proc. ACM MobiHoc*, 2005, pp. 68–77.

[28] A. P. Subramaniam, H. Gupta, and S. R. Das, "Minimum-interference channel assignment in multi-radio wireless mesh networks," in *Proc. IEEE SECON*, 2007, pp. 481–490.

[29] J. So and N. Vaidya, "Multi-channel MAC for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *Proc. ACM MobiHoc*, 2004, pp. 222–233.

[30] P. Kyasanur and N. Vaidya, "Routing and link-layer protocols for multichannel multi-interface ad hoc wireless networks," *Comput. Commun. Rev.*, vol. 10, no. 1, pp. 31–43, 2006.

[31] M. Yun, Y. Zhou, A. Arora, and H.-A. Choi, "Channel-assignment and scheduling in wireless mesh networks considering switching overhead," in *Proc. IEEE ICC*, 2009, pp. 1–6.

[32] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large scale video-on-demand systems," in *Proc. ACM EuroSys*, 2006, pp. 333–344.

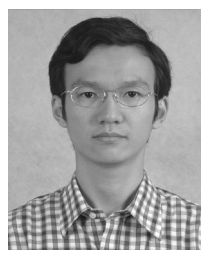
[33] T.-C. Chiueh, A. Raniwala, R. Krishnan, and K. Gopalan, "Hyacinth: An IEEE 802.11-based multi-channel wireless mesh network," 2005 [Online]. Available: http://www.ecsl.cs.sunysb.edu/multichannel/

[34] "Video trace library," 2011 [Online]. Available: http://trace.eas.asu.edu/

[35] P. Seeling, M. Reisslein, and B. Kulapala, "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial," *IEEE Commun. Surveys Tutorials*, vol. 6, no. 3, pp. 58–78, 3rd Quart., 2004.

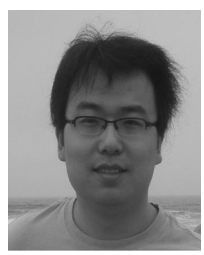
[36] G. V. der Auwera, P. T. David, and M. Reisslein, "Traffic and quality characterization of single-layer video streams encoded with the H.264/MPEG-4 advanced video coding standard and scalable video coding extension," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 698–718, Sep. 2008.

[37] G. V. der Auwera and M. Reisslein, "Implications of smoothing on statistical multiplexing of H.264/AVC and SVC video streams," *IEEE Trans. Broadcast.*, vol. 55, no. 3, pp. 541–558, Sep. 2009.



Yong Ding received the B.S. and M.S. degrees in computer science from Southeast University, Nanjing, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer science from Michigan State University, East Lansing, in 2010.

His research interests are in the areas of distributed systems and computer networking, including wireless sensor networks, vehicular ad hoc networks, and wireless mesh networks.



Yang Yang received the B.S. and M.S. degrees in automation from the University of Science and Technology of China, Hefei, China, in 2004 and 2007, respectively, and the Master's degree in computer science from Michigan State University, East Lansing, in 2010.

His research interests are in the areas of distributed and networking systems, wireless mesh networks, P2P networks, and video-on-demand systems.

Li Xiao (M'00–SM'10) received the B.S. and M.S. degrees in computer science from Northwestern Polytechnic University, Xi'an, China, and the Ph.D. degree in computer science from the College of William and Mary, Williamsburg, VA, in 2002.

She is an Associate Professor of computer science and engineering with Michigan State University, East Lansing. Her research interests are in the areas of distributed and networking systems, overlay systems and applications, and wireless sensor and mesh networks.