An Overlay Subscription Network for Live Internet TV Broadcast

Ying Cai, *Member*, *IEEE*, and Jianming Zhou

Abstract—We propose a framework, called *Overlay Subscription Network* (OSN), for live Internet TV broadcast, where a subscriber can choose to watch at any time. This framework allows the source server to incrementally build a topology graph that contains the network connections not only from the server to each subscriber, but also among the subscribers themselves. With such a topology graph in place, we consider efficient overlay multicast for scalable OSN services. We first show that idling nodes, which do not receive video data for their own playback, can actually be used for data forwarding to significantly reduce the cost of overlay multicast. In light of this observation, we then propose a novel overlay multicast technique that distinguishes itself from existing schemes with these three aspects. First, the proposed technique is centered on the topology graph and can take advantage of the actual network connections among the subscribing nodes. Second, the new scheme is able to find and incorporate *appropriate* idling nodes in multicast to reduce network traffic. Third, with our approach, a node can be used in multiple multicast trees for data forwarding to improve the overall system performance. We evaluate the performance of the proposed technique through simulation. Our extensive studies show that the proposed framework has the potential to enable the Internet, a vehicle up to date mainly for transferring text and image data, for large-scale and cost-effective TV broadcast.

Index Terms—Overlay subscription networks, overlay multicast, video services, live streaming.

1 Introduction

Unlike text or image files, streaming a video to a remote client takes a significant amount of communication bandwidth. A video server typically can sustain only a very limited number of concurrent video streams. This problem, known as *server or network-I/O bottleneck*, limits the scalability of video services. To improve server throughput, two main categories of techniques have been proposed. The techniques in the first category explore the facility of IP multicast for clients to share server bandwidth. Many early techniques, such as *on-demand multicast* [8], [17], [13], [11] and *periodic broadcast* [26], [18], [16], are in this category. The techniques in the second category are often referred to as *overlay multicast* [23], [6], [22], [7]. Instead of relying on IP multicast, overlay multicast expands the server capacity by requiring the clients which are being served to buffer and forward their incoming video streams to serve others.

Overlay multicast is an attractive solution for video distribution over today's Internet, where the deployment of IP multicast has been slow and difficult due to issues like group management, congestion and flow control, and security [9]. Unlike the traditional client/server architecture, this strategy allows a client to contribute its computing resource to serve the entire community, rather than just being a burden to some central server. Under this approach, the clients, together with the source server, form a live video distribution tree that dynamically expands and shrinks as clients join and leave. Since the clients typically access the Internet from vastly different network domains, the problem of server bottleneck can be addressed effectively by arranging video data to flow through different network links to reach the receiving ends.

Suppose a television broadcast company wants to stream its real-time TV programs over the Internet to its subscribers. Similarly to traditional satellite/cable services, these subscribers pay a monthly fee and register their computers to watch the programs. In this paper, we refer to the network formed by the source server and its subscribers as the *Overlay Subscription Network* (OSN). Such a network has an inherent feature that makes it possible to apply the concept of overlay multicast. That is, an OSN is a *trusted* overlay network in the sense that its subscribers trust the source server and, hence, pay for its services. Because of such trustiness, effective incentive mechanisms can be designed to encourage a subscriber to contribute its resource to serve others. As a simple example, a member can be given some discount on its subscription fee based on the amount of data it forwards. Such a realistic monetary incentive can effectively turn many subscribers into service partners to assist in data forwarding, a prerequisite to applying overlay multicast in reality. Effective incentive mechanisms for peer-to-peer networks have recently attracted great research interests; interested readers are referred to [15], [21], [4], [12]. In this paper, we simply assume some incentive mechanism is in place to motivate subscribing clients to contribute their resources in video services.

In this paper, we consider overlay multicast for large-scale Internet TV broadcast. Because of the inability of existing technologies, traditional cable/satellite broadcasts are still the primary media for distributing continuous and endless video programs such as TV broadcast. The main contributions of this paper are as follows:

• *The authors are with the Department of Computer Science, Iowa State University, Ames IA 50011. E-mail: {yingcai, jmzhou}@cs.iastate.edu.*

Manuscript received 10 Sept. 2005; revised 24 Apr. 2006; accepted 8 July 2006; published online 18 Oct. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0359-0905.

1. We propose a framework by which the server can incrementally learn and build the topology graph that reflects the actual network connections from the server to each subscriber and among the subscribers themselves. Such topology information is crucial to constructing efficient overlay multicasts. Existing topology-oriented techniques, such as TAG [19], [20], consider only the network connections from the server to each individual client.
2. An OSN may have a large number of subscribing nodes, yet, at any one time, only a small percentage of them are actually receiving the programs for playback. We demonstrate that recruiting the idling nodes, which are online but not receiving the programs, for data forwarding can significantly reduce the cost of overlay multicast. We propose a novel technique that is able to find and incorporate appropriate idling nodes for data forwarding. Such capability is unique since existing techniques can leverage only the computing resource of the clients who are playing back videos themselves.
3. The source server may provide multiple sessions of video programs simultaneously. Observing the advantages of leveraging idling nodes, we consider how a node can be recruited in multiple sessions to further reduce network traffic. This capability is also unique, considering the fact that, in existing techniques, a client can only contribute its resource to serve other clients who are in the same session. Because of this limitation, the server needs to build a distinct multicast tree for each session.

The remainder of this paper is organized as follows: In Section 2, we discuss the OSN framework and its management in details, including subscription management, topology graph, and stream management. We present the concept of incentive forwarding in Section 3 and then in Section 4, apply this idea in constructing single- and multisession overlay multicast. The performance of the proposed technique is evaluated in Section 5. In Section 6, we give our concluding remarks.

2 Overlay Subscription Network (OSN)

OSN is an overlay network consisting of one central server that provides continuous video programs (e.g., TV broadcast) and a number of subscribers that register their receivers and pay a monthly fee to watch the programs. Without causing ambiguity, we will use the terms subscriber, member, and node interchangeably. In OSN, the server maintains a *subscription database* storing all information about its subscribers, including ID and password, IP address, payment, incentive, and so on. The server may ask a member to forward video data to serve others. If a member agrees to serve others, the server will record and store in the database the time duration and bandwidth contributed by the member. We assume some mechanism is used to calculate incentive amount based on the contribution of a member within one billing cycle.

OSN is a registration-based network. This makes it possible to use a *topology graph* to record its underlying network topology. When a new member subscribes to the system, the server detects its path to the member and adds the path in the topology graph. For path finding, many approaches can be used. A simple approach is using *tracepath* [1], which is ICMP-based and has been used extensively for Internet topology discovery. The path obtained may be at the router level or the coarse-grained AS level. Without loss of generality, we assume the router-level path. If only AS information is available, we treat each AS as a router. In addition to the path from the server to each subscriber, the topology graph can also store the routing paths among the subscribers, when such information becomes available. That is, when a subscriber A forwards its incoming stream to serve another subscriber B, A can detect and report the actual streaming path to the server, which will update the topology graph if necessary.

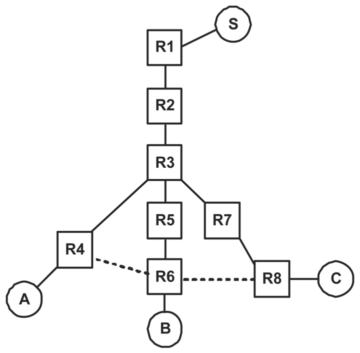


Fig. 1. Topology graph.

In this paper, we argue that a detailed and accurate topology graph is crucial to constructing efficient overlay multicast. Although exploiting network topology was first investigated in [19], [20], the proposed *Topology Aware Grouping* (TAG) technique considers only the paths from the server to its clients. In particular, it assumes that only the network links in these paths can be used by the clients to communicate with each other. Such an assumption can be unrealistic. As an example, consider Fig. 1. It shows a source server S, three clients A, B, and C, and the paths from the server to each of them. Suppose S is serving A and B when C arrives. In TAG, the server will ask A to forward its video data to serve C and expects the traffic to flow through router R_4, R_3, R_7, and R_8. In reality, however, the actual streaming path may be quite different. For instance, the stream from A to C may actually go through R_4, R_6, and R_8, the links of which are unknown to the server in TAG. In general, a topology containing only the path from the server to each client can be too coarse to be relied upon in building an efficient overlay multicast. In OSN, as the server learns the actual connections among its subscribers, it can incrementally update the topology graph for better multicast construction. In the above example, A is required to report its actual streaming path to the server. With the newly discovered links, the server may later ask B to serve C.

In addition to the subscribers and their topology information, the source server also tracks all streams and their streaming paths. A stream from node X to Y is denoted as $X \Rightarrow Y$ if the stream flows from X to Y *directly*.

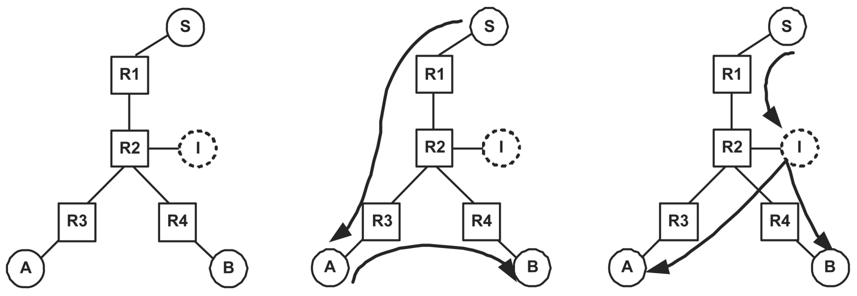


Fig. 2. Motivation example.

Suppose S in Fig. 1 serves A directly and A forwards its stream to B. Then, two streams are formed, $S \Rightarrow A$ and $A \Rightarrow B$. We say a router is an *active* router if there is at least one stream flowing through it. Thus, given a stream, all routers on its streaming path are active. The information of each stream and its streaming path can be stored in a binary relation, each row being a tuple of $(X, Y, \mathcal{R})$, denoting that stream $X \Rightarrow Y$ flows through router $\mathcal{R}_i$. Thus, given a router, we can find the streams, if any, flowing through it. Many access structures can be used to support such queries efficiently. For example, we can hash or build a B^+-tree index on the network links. Alternatively, we can also store the entire information in an adjacency matrix instead of a relational table. Hence, we will not concern ourselves with its implementation details.

3 Incentive Forwarding

Similar to traditional TV broadcast, the programs provided by OSN are continuous and endless—24 hours a day, 7 days a week. A SON may have a large number of registered nodes, yet, at any one time, only a small percentage of them are actually watching, while many of the rest are just idling. We refer to these two types of nodes as *playing* nodes and *idling* nodes, respectively. We note that, unlike regular TV sets, which are dedicated to broadcast receiving and could be turned off when they are not in use, the computers registered in an OSN are more likely left powered on even though they are not receiving a broadcast. These computers may be powered on for other purposes, such as e-mail/document processing. In fact, according to [2], many office and residence computers are simply never turned off.

With realistic monetary incentive in place, a subscribing node can be highly motivated in data forwarding. This makes it possible to apply overlay multicast for scalable broadcast in OSN. Efficient overlay multicast has been studied intensively in the past few years and many approaches have been proposed, such as *Chaining* [23], *Narada* [6], [5], *NICE* [22], *ZIGZAG* [24], and *TAG* [19], just to name a few. Existing techniques, however, consider only the playing nodes in constructing overlay multicast. That is, they can leverage only the computing resource of playing nodes for data forwarding. In this paper, we argue that idling nodes, when chosen appropriately, can be used to significantly reduce the cost of overlay multicast.

As a motivation example, consider Fig. 2. It shows a source server S and three subscribing nodes, A, B, and I, and their underlying network topology. Suppose A and B are playing and I is idling. If the server serves A and B directly, the network traffic on links R_1 and R_2 will be duplicated. Alternatively, the server may serve A directly and ask A to serve B. This approach creates duplicate traffic on links R_2 and R_3. In addition, the data arriving at B experiences a longer latency since it flows through $S \rightarrow R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow A \rightarrow R_3 \rightarrow R_2 \rightarrow R_4 \rightarrow B$. Similar problems exist if S serves B directly and asks B to serve A. Now, suppose I, an idling node, has the capacity and can be recruited to serve A and B. Then, the server can send data to I first and let I forward to A and B. Apparently, this approach minimizes the backbone network traffic and also ensures good data freshness.

When an idling node is recruited to forward data, we say this node becomes an *incentive node*.[1] A major challenge of implementing such incentive forwarding is to find and incorporate *appropriate* idling nodes in constructing overlay multicast. Unlike a playing node, an idling node does not need video data for its own playback. Thus, an idling node should be recruited only when its assistance in data forwarding can reduce the service cost. Obviously, such an idling node must be able to forward its incoming stream to serve at least two other nodes.

4 Session Management

In this section, we consider the problem of constructing overlay multicast in OSN. Our proposed techniques are centered on the topology graph discussed earlier. To facilitate our discussion, we define the following terms and notations:

- $Path(X, Y)$ denotes the sequence of routers on the shortest path from nodes X to Y and $Hop(X, Y)$ the number of routers on $Path(X, Y)$. In Fig. 2, $Path(S, A) = \mathcal{R}_1 \rightarrow \mathcal{R}_2 \rightarrow \mathcal{R}_3$ and $Hop(S, A) = 3$.
- Given a router $\mathcal{R}$, $Ring(\mathcal{R}, i)$ denotes the set of routers that are an i-hop away from $\mathcal{R}$, where $i \geq 0$. In Fig. 2, $Ring(\mathcal{R}_2, 0) = \{\mathcal{R}_2\}$ and $Ring(\mathcal{R}_2, 1) = \{\mathcal{R}_1, \mathcal{R}_3, \mathcal{R}_4\}$.
- Given a node and a router that connect each other directly, we say the node is the router's *local node* and the router is a node's *local router*.

1. A playing node is also paid if it is recruited for data forwarding. Thus, in terms of monetary incentive, such a playing node is also an incentive node.

- A node's *capacity* is the maximum number of children it can have and its *degree* is the number of children it is serving.

In the following discussion, we first present our technique for single-session management, where we assume that the server supplies only one channel of video programs. We will then extend our technique for multi-sessions management. In this case, the server broadcasts a number of channels simultaneously, similarly to the traditional TV broadcast.

4.1 Single-Session Management

In OSN, a node can be offline or online. An online node is *idling* if it does not receive any video stream. Otherwise, the node must participate in some multicast session and is called an *active* node. An active node can be in *playing* or *incentive* mode. In the former case, the node plays back the stream it receives, while, in the latter case, the node is an idling node and is recruited as an incentive node for data forwarding. Since only the server can decide when to recruit an incentive node, a node can make itself only offline, idling, or playing. Assuming that the server provides only one session of video programs, we discuss in the following how to handle when a node changes its status.

4.1.1 A Node Becomes Online

When an offline node N becomes online, it notifies server S. If N can serve at least two children and at least two streams flow through N's local router, N can be recruited as an incentive node to save the network traffic. In this case, the server calls the following $Bundle(N, SS)$ procedure, where SS denotes the set of streams flowing through N's local router:

```
$Bundle(N, SS)$
1. Check each stream in $SS$ and find the one, say $X \Rightarrow Y$, that has the smallest $Hop(X, N)$.
2. Create two new streams, $X \Rightarrow N$ and $N \Rightarrow Y$, and terminate stream $X \Rightarrow Y$.
3. Repeat the following until $N$ can have no more child or $SS$ becomes empty:
    • Check each stream in $SS$ and find the one, say $X \Rightarrow Y$, that has the largest $Hop(X, N)$.
    • Create a new stream $N \Rightarrow Y$ and terminate $X \Rightarrow Y$.
```

When a stream $X \Rightarrow Y$ is bundled, the traffic saved can be calculated as $Hop(X, N) * b$, where b is the playback rate of the stream. We use Fig. 3 to explain the above bundle algorithm. Suppose N can serve two children and three streams, $X_1 \Rightarrow Y_1$, $X_2 \Rightarrow Y_2$, and $X_3 \Rightarrow Y_3$, flow through N's local router $\mathcal{R}$. To minimize the cost of including N in the session, the server chooses X_1 to be N's parent since X_1 is closest to N. Thus, X_1 serves N and N forwards its stream to Y_1, which was X_1's child. Since N has the capacity for one more child, Y_3 becomes its child. Y_3 is chosen because the path from X_3 to N is the longest, thus maximally reducing the network traffic.

Note that, whenever a stream $X \Rightarrow Y$ is terminated, the server needs to check if X can be removed from the session.

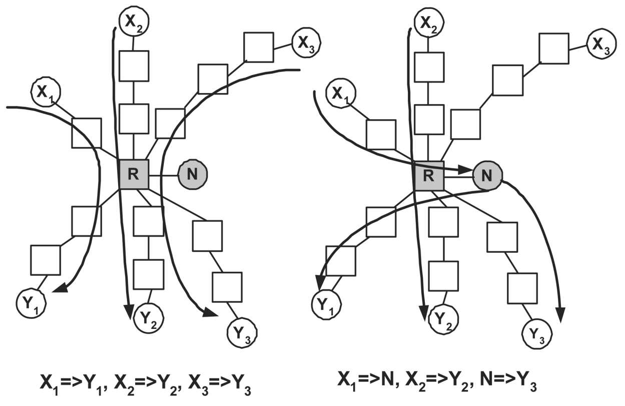


Fig. 3. Stream bundling.

If X is an incentive node and has no more children, the server removes X from the session by terminating the stream from X's parent to X, and then recursively checks if X's parent can be removed.

4.1.2 A Node Becomes Playing

When a node N wishes to join the session, it notifies the server. If the node is currently an incentive node, i.e., is being recruited in the session to serve others, the server simply updates the node's status as playing and the node can start to playback the stream. Otherwise, the server includes the node in the session as follows: In our discussion, we first assume node N's capacity is 0 and then extend our algorithm to handle when this is not true.

To include N in the session, the server first checks if there is any active router on $Path(S, N)$. If none of them is active, the server starts a new stream $S \Rightarrow N$ to serve N directly. Otherwise, the server tries to find an existing active node or recruit a new incentive node to serve N. Let $\mathcal{R}_j$ be the active router on $Path(S, N)$ that is closest to N. Since $\mathcal{R}_j$ is the nearest router on $Path(S, N)$ that N can access to some on-going stream, N's parent should be as close as possible to $\mathcal{R}_j$ in order to minimize the cost of serving N. The server uses the following procedure to find a list of parent candidates for N:

```
$FindParentCandidates(\mathcal{R}_j)$
1. Set $CandidateSet = \emptyset$ and $i = 0$.
2. While $CandidateSet == \emptyset$, do the following:
    • Set $LocalNodeSet = \emptyset$.
    • For each router on $Ring(\mathcal{R}_j, i)$, add its local nodes to $LocalNodeSet$.
    • Check each node in $LocalNodeSet$ and add it to $CandidateSet$ if the node is currently active and can serve one more child.
    • If $CandidateSet \neq \emptyset$, return.
    • Otherwise, check each node in $LocalNodeSet$ and add it to $CandidateSet$ if the node is currently idling and can serve at least two children.
    • Increase $i$ by 1.
```

In the above algorithm, the server first checks router $\mathcal{R}_j$ (i.e., $Ring(\mathcal{R}_j, 0)$) to see if any local active node can serve N.

If no such active node is available, it tries to recruit a new incentive node. If this also fails, it expands the search scope by checking $Ring(\mathcal{R}_j, 1)$, $Ring(\mathcal{R}_j, 2), \ldots,$ until at least one parent candidate is found. Apparently, such a *ripple* searching process will locate N's nearest parent candidates.

Suppose the server stops its search on $Ring(\mathcal{R}_j, i)$. If the candidates found are active nodes, the server can simply choose the one that has the minimum latency to serve N. Otherwise, these candidates must be idling nodes and the serve recruits one of them as a new incentive node. Before discussing how to select a new incentive node, we first discuss the cost of including an incentive node, say I, in the session. Given a stream $X \Rightarrow Y$ flowing through some router on $Ring(\mathcal{R}_j, i)$, we can redirect the stream to serve I, i.e., $X \Rightarrow I$ and $I \Rightarrow Y$. Thus, the cost of serving I is $Hop(X, I) + Hop(I, Y) - Hop(X, Y)$. Since I is recruited to serve N, the total cost of including both I and N, which we will denote as $Cost(X \Rightarrow Y, I, N)$, is equal to $Hop(X, I) + Hop(I, Y) + Hop(I, N) - Hop(X, Y)$. This can be seen as the cost of recruiting I to serve N. Thus, given a set of streams flowing through $Ring(\mathcal{R}_j, i)$, we can find out the one that can be redirected to serve I and N with the minimal cost. This node is then recruited as an incentive node. A more formal description of such a selection process is given below. Note that, after selecting a node, the server needs to contact the node to find out if it is online and can be used for incentive forwarding. If not, the server repeats the above process for another candidate.

```
$ChooseIncentiveNode(CandidateSet, N)$
1. $SS = \emptyset$.
2. For each active router $\mathcal{R}$ on $Ring(\mathcal{R}_j, i)$, add all
   streams that flow through $\mathcal{R}$ to $SS$.
3. For each node $I$ in $CandidateSet$ and each stream
   $X \Rightarrow Y$ in $SS$, calculate $Cost(X \Rightarrow Y, I, N)$.
4. Return $I$ if $Cost(X \Rightarrow Y, I, N)$ is the smallest.
```

Our discussion so far assumes that N cannot have any child. If N's capacity is not 0, it can be included in the session as follows: The server first searches $Ring(\mathcal{R}_j, 0)$, trying to find an active node or recruit a new incentive node to serve N. If this fails, the server continues to search $Ring(\mathcal{R}_j, 1)$, $Ring(\mathcal{R}_j, 2), \ldots,$ and so on. However, it stops once the search is expanded to the ring that contains N's local router. Since N can serve at least one child, it can be included in the session by redirecting a stream that flows through the routers on the ring. If more than one stream is available, the server chooses the one with the least cost.

4.1.3 A Node Becomes Offline or Idling

A playing node or an incentive node may decide to be idling or offline. It is also possible that a node may wish to reduce the number of its current children. When this happens, the server schedules an emergent stream to serve each affected child. At the same time, the server uses the algorithms discussed in the previous section to either find an existing active node or recruit an incentive node to serve the child. Once a new parent is found for a child, the emergent stream can be terminated.

A node leaving from the session may cause temporal service disruption to its downstream nodes. To avoid such an undesired effect, one can use multiple descriptions coding to encode a video stream into a set of substreams and, for each substream, build a multicast tree using our proposed technique. A node can then receive the set of substreams from different parents and recover the original stream even if some parent fails. Much effort has been done on fault-tolerant overlay multicast; interested readers are referred to [25], [14], [10], [27] for more information. In OSN, incentive mechanisms can also be used to encourage a node to become a stable service partner or, before leaving the session, give some grace period for smooth service transition. In addition, the server can build some reputation management and choose only the subscribers with a good reputation for data forwarding. These subjects are beyond the scope of this paper and we will leave them for future study.

4.2 Multisession Management

A TV station normally broadcasts a number of channels simultaneously. Likewise, an OSN may provide multiple sessions of video programs and a subscriber can choose to watch any one of them at any time. In this section, we discuss how our single-session management technique can be extended for multisessions broadcast. To save network traffic, an idling node can actually be incorporated in multiple sessions, subject to its capacity (e.g., downloading and uploading bandwidth). With existing techniques, the server needs to build a *distinct* multicast tree for each session. In these schemes, only the playing nodes can contribute their resources and only the nodes in the same session can serve each other.

As discussed early, when an offline node N becomes online, it may be included as an incentive node to bundle the streams flowing through its local router $\mathcal{R}$. In the case of multisession broadcasts, these streams may belong to different sessions. Since N is an idling node, it can be recruited in any session for data forwarding. The question is, which sessions should N be used for? Recall that, when a stream $X \Rightarrow Y$ is bundled, the traffic saved can be calculated as $Hop(X, N) * b$, where b is the playback rate of the stream. Thus, we can calculate the savings of bundling each session of streams and select the session with the maximum savings to bundle first. This process is repeated until either all sessions are bundled or N runs out of its idling capacity. Note that only the sessions with at least two streams can be bundled and to bundle one session, N needs to have bandwidth to download one stream and forward it to serve at least two children.

Let N be a node that wants to watch session S_i. If N is a playing node and/or an incentive node that is currently serving some *nonpeer* nodes (i.e., not in session S_i), then the server may need to remove these children and find them new parents. This process, which we refer to as *sanitization*, is necessary for two reasons. First, serving nonpeer nodes may exhaust N's capacity and make N unable to download data from its own session. For example, N may run out of its downloading bandwidth. Second, it is preferable for a playing node to serve other nodes that are in the same session. Serving its *peer* nodes does not require the playing node to have extra bandwidth for data downloading. When a playing node has extra capacity that is not used by its own session, it is possible to recruit this node as an incentive node for other sessions to save their traffic. However, this node must be sanitized again when it can be used to serve the nodes in its own session (e.g., a new peer node joins). To avoid such overhead, we can limit a playing node to join only one session, even if some of its capacity may not be in use for some time period. After N is sanitized, the server can use the algorithms presented in the previous section to find N a parent. Note that, in the step of *FindParentCandidate*, a node should now be considered as an idling node as long as it is not a playing node and has sufficient idling capacity.

5 Performance Study

To evaluate the performance of our proposed techniques, we have implemented a detailed OSN simulator that can provide a number of current TV programs over the Internet. For performance comparison, we implement three different overlay multicast techniques:

- *OSN_BASE*: In this scheme, only the playing node can contribute their resource to serve others. This approach is similar to *TAG* [19], [20], an existing topology-oriented overlay multicast technique. However, OSN_BASE is centered on the topology graph and can take advantage of the actual network connections among the subscribers.
- *OSN_SINGLE*: This approach implements the proposed single-session management technique. It is able to find and incorporate appropriate idling nodes for data forwarding. However, it allows a node to participate in one session only. In OSN_SINGLE, a distinct multicast tree is built for each session.
- *OSN_MULTIPLE*: This technique is the implementation of the proposed multisessions management technique. In OSN_MULTIPLE, a node can participate in a number of sessions, subject to its available capacity.

For simplicity, we will refer to the above three techniques as *BASE*, *SINGLE*, and *MULTIPLE*, respectively. Similarly to other real-time overlay multicast techniques [6], [22], [19], we choose these performance metrics:

- *Link Stress*: Given a network link, its stress is defined to be the total number of streams flowing through it. For each network link, we calculate its stress and report their sum. This metric is also referred to as the network cost of overlay multicast.
- *Maximum Link Stress*: This metric is defined to be the maximum stress of all links in a multicast tree. A higher value of this metrics means a higher chance of creating a network bottleneck.
- *Mean Relative Delay (MRP)*: Assuming the stream serving node X flows through n routers from source S before arriving at X, the relative delay for X is defined to be $\frac{n}{m}$, where m is the number of routers in the shortest path from S and X. This metric reflects the relative increase of packet delay as a result of using overlay forwarding. We compute the relative delay for all playing nodes and report their mean value.

TABLE 1
Parameters

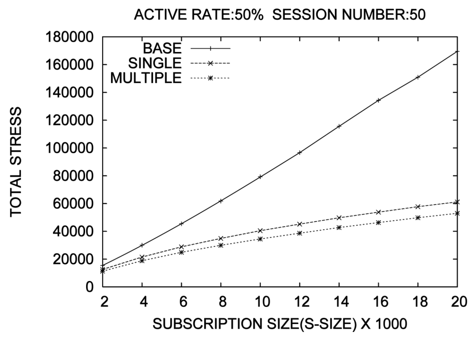
Parameter	default	variation
subscription size	10,000	2,000 – 20,000
active rate	50%	5% – 100%
session number	50	10 – 100

For SINGLE and MULTIPLE, we also report the number of incentive nodes used in these two approaches.

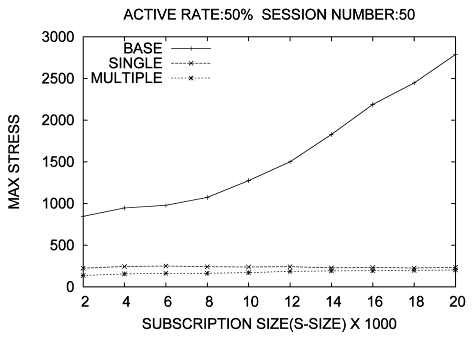
We are interested in how the above performance metrics are affected by *subscription size* (i.e., the number of subscribers), *active rate* (i.e., the percentage of the subscribers actually watching), and *session number* (i.e., the number of video sessions provided by the server). In our simulation, the underlying Internet topology is created using *Brite*'s TOP_DOWN model [3] and consists of 5,000 routers. The topology is a two-level network hierarchy, interconnected higher level (AS level) and lower level stub domains (router level), and each AS contains 20 routers in average. We then generate a number of end systems (i.e., subscribers) and randomly connect them to the edge routers within each AS. The bandwidth of a subscribing node is randomly set to be in between one and six streams. We assume symmetric network connections and do not distinguish downloading and uploading bandwidth. Roughly, we simulated a local TV broadcast station of a small city. Each data point in the performance figures is averaged from 20 simulation runs, each randomly choosing one multicast source. The confidence level is 98 percent. Table 1 summarizes the parameter values used in our simulation.

5.1 Effect of Subscription Size

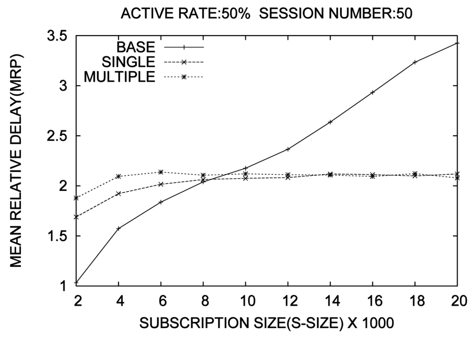
In this study, we varied the number of subscribers from 2,000 to 20,000 and assume 50 percent of them are playing. The number of video sessions provided by the server is 50. The performance results are plotted in Fig. 4. It shows that, under all scenarios, BASE incurs the highest total stress and max stress. In particular, the performance gap between BASE and the other two schemes increases sharply as the number of subscribers increases. This result confirms that leveraging idling nodes for data forwarding can indeed significantly reduce network traffic and minimize the chance of creating network bottleneck. As for MULTIPLE and SINGLE, Figs. 4a and 4b both show that MULTIPLE consistently outperforms SINGLE in terms of total stress and max stress. By allowing an idling node to be recruited in multiple sessions, MULTIPLE has a better chance than SINGLE in locating a nearest parent for a new playing node. We now look at the mean relative delay caused by the three techniques. Fig. 4c shows that, when the subscription size is small, BASE has the least delay. BASE leverages only the playing nodes for data forwarding. When the number of playing nodes is smaller, their relative distance is larger as



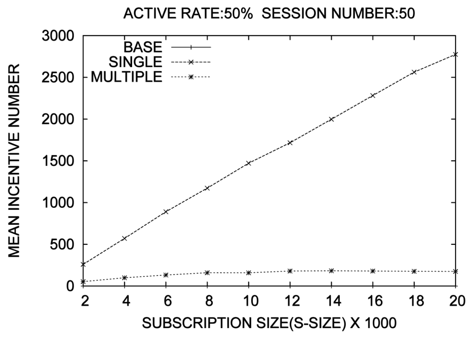
(a)



(b)



(c)



(d)

Fig. 4. Effect of subscription size.

they randomly span in the entire network. As a result, the chance is higher for the server to serve each individual subscriber. As Fig. 4c shows, when the number of subscribers is 2,000, the mean relative delay is nearly 1 for BASE, indicating that almost all playing nodes are served directly by the server. Fig. 4b confirms that, in this setting, BASE has presented a major network bottleneck, where the worst network link has to sustain more than 750 video streams. In contrast, both SINGLE and MULTIPLE have quite stable relative delay and max stress. As Figs. 4b and 4c show, the two techniques are not sensitive to the number of subscribers in terms of max stress and relative delay. This feature is highly desirable and it indicates they can be used in a large-scale OSN with expected good performance. Fig. 4d shows that SINGLE uses much more incentive nodes than MULTIPLE. This indicates that, when a new playing node comes, SINGLE has to find the node a farther parent, although many times a nearby incentive node could be recruited.

5.2 Effect of Active Rate

In this study, we fixed the number of subscribers at 10,000 with 50 sessions and varied the active rate from 10 percent to 100 percent. The performance results are plotted in Fig. 5. Figs. 5a and 5b show that the total stress and max stress incurred by BASE are both the highest and they become worse and worse than the other schemes as the active rate increases. This is a very interesting phenomenon. Given a fixed number of subscribers, increasing the active rate increases the number of playing node and reduces the number of idling nodes. When the active rate becomes 100 percent of active rate, all three schemes have exactly the same number of playing nodes for SINGLE and MULTIPLE, there is no idling node to recruit at all. Thus, one may expect all three schemes to eventually perform similarly. However, Figs. 5a and 5b show that BASE remains the worst performer. Such a performance difference is due to the different orders of adding nodes in sessions. In SINGLE and MULTIPLE, an idling node is recruited in a session whenever doing so reduces the network traffic. When an incentive node later becomes a playing node, there is no extra cost since the node has already been in the session. Thus, SINGLE and MULTIPLE are flexible in the order of adding nodes. In contrast, BASE includes a node in a session only when it becomes a playing node. When the active rate is low, the server may have to find a parent node very far away to serve a new playing node. As the active rate increases, more and more idling nodes become playing nodes. However, these idling nodes may again have to find their parents, which are far away because their nearby playing nodes have run out of their capacity in order to serve early playing nodes. This performance result indicates that in the application scenarios where recruiting idling nodes is infeasible, periodically reconstructing the overlay multicast as new playing nodes join can significantly improve the overall system performance. Fig. 5a shows that, when the active rate is low, BASE incurs the least mean relative delay. As we have explained in the previous study, this is simply because, when the number of playing nodes is low and their relative distance is large, most of them are served directly by the server. In this simulation, MULTIPLE again consistently outperforms SINGLE in all settings.

5.3 Effect of Session Number

In this study, we fixed the number of subscribers at 10,000 within a network with a total of 5,000 routers. We assume

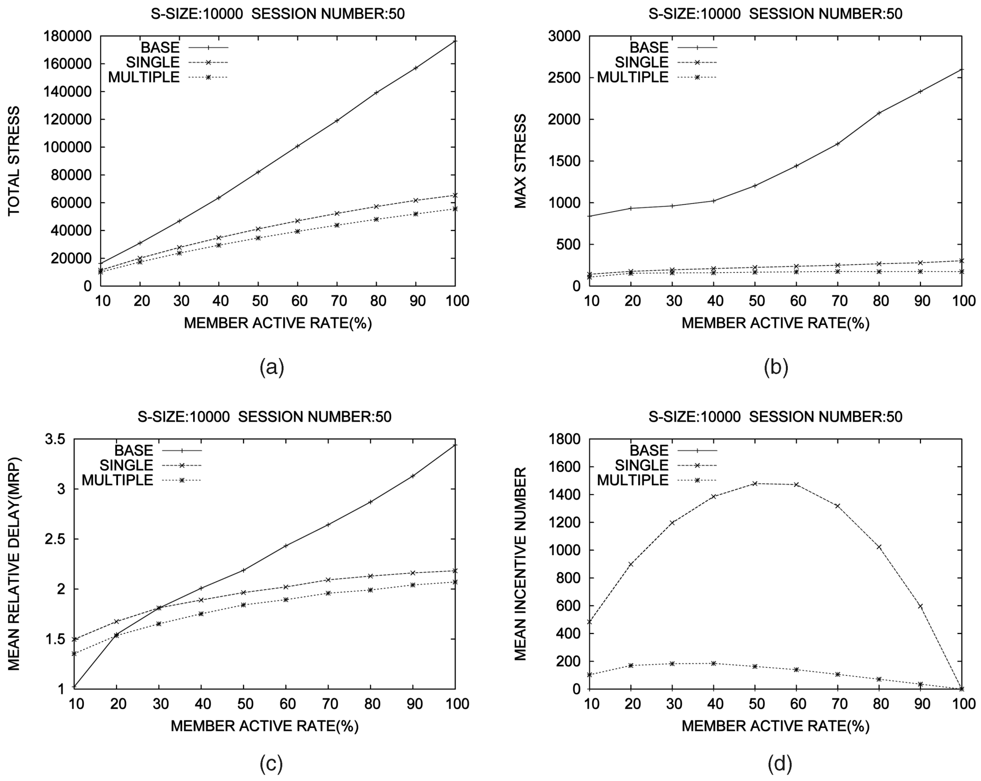


Fig. 5. Effect of active rate.

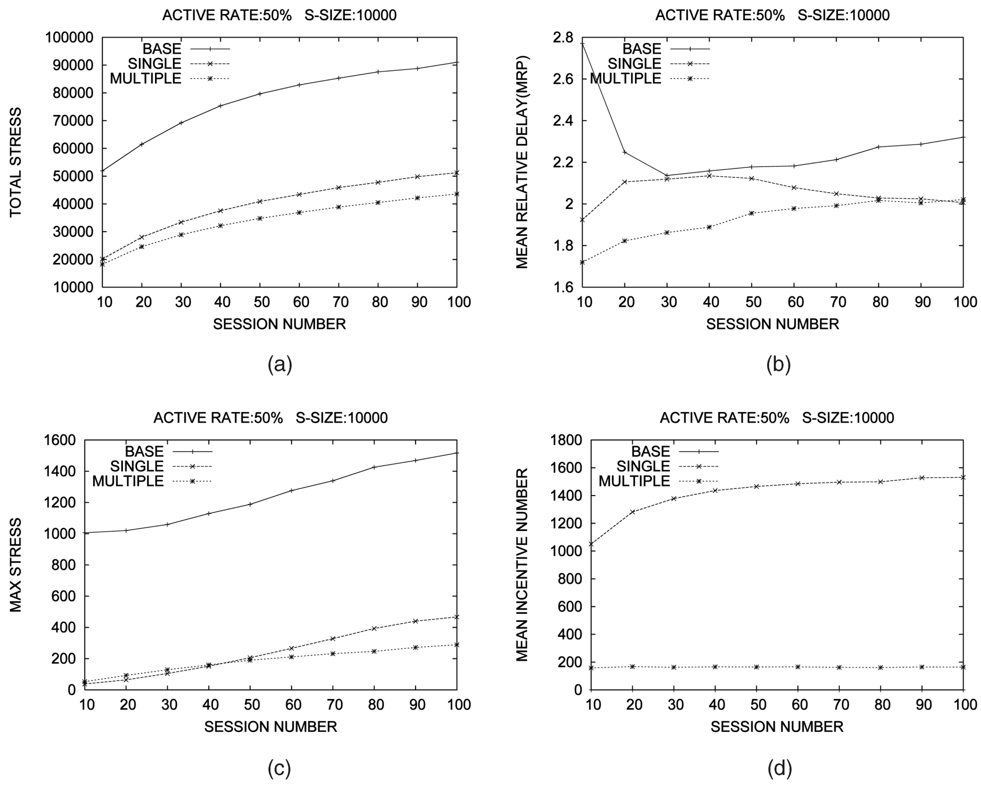


Fig. 6. Effect of session number.

the active rate is fixed at 50 percent and vary the session number from 10 to 100. The performance results are plotted in Fig. 6. Similarly to the previous two studies, the results show that BASE imposes much more network traffic and results in a longer relative delay than the other two approaches that leverage the idling capacity of the subscribers. As the session number increases, MULTIPLE outperforms SINGLE more and more. This is due to the fact that, in SINGLE, once an idling node is recruited in one session, the node cannot be used in other sessions even if it has sufficient idling capacity. As a result, the server would need to find a farther parent for a new playing node. As the number of sessions increases, the chance of finding a nearer parent is reduced. Fig. 6c shows that, when the session number is low, MULTIPLE incurs more max stress than SINGLE. Since an idling node is used to support more sessions, MULTIPLE generates more network traffic on the routers that are near to the incentive node. However, as the session number increases, MULTIPLE gradually outperforms SINGLE. With a higher session number, the chance is higher for SINGLE to have to find a farther parent to serve a new playing node. The network traffic from the parent to the new playing node then needs to flow more network links. As Fig. 6d shows, more and more incentive nodes are recruited in SINGLE when the session number increases.

6 Concluding Remarks

Because of the inability of existing technologies, traditional cable/satellite broadcasts are still the primary media for distributing TV programs. In this paper, we propose a framework, namely, *OSN*, aiming at enabling the Internet for large-scale and cost-effective TV broadcast. Similarly to cable/satellite broadcasts, the video programs provided by OSN are continuous and endless—24 hours a day, 7 days a week. There may be a large number of subscribers, yet, at any one time, only a small percentage of them are actually watching. However, unlike regular TV sets, which are dedicated to broadcast receiving and could be turned off when they are not in use, the computers registered for Internet TV services are more likely left powered on even though their owners are not watching. These computers may be in use and, thus, powered on for other purposes, such as e-mail/document processing. In fact, many office and residence computers are powered on without actually being in use [2]. In this paper, we show that such idling nodes can be used in constructing highly efficient overlay multicast. Since an idling node does not need video data for its own playback, it should be incorporated in multicast only when doing so can reduce the cost of overlay multicast. To find and incorporate only the appropriate idling nodes for data forwarding, we propose a novel topology-oriented overlay multicast technique. Our extensive simulation studies have showed convincingly that leveraging idling nodes for data forwarding can result in significant performance advantages in terms of reducing network traffic and balancing the workload of network links.

References

[1] ftp://ftp.inr.ac.ru/ip-routing/iputils-current.tar.gz, 2005.

[2] http://www.cquest.utoronto.ca/env/env421h/energy/computers.html, 2005.

[3] http://www.cs.bu.edu/brite/download.html, 2005.

[4] C. Buragohain, D. Agrawal, and S. Suri, "A Game Theoretic Framework for Incentives in P2P Systems," *Proc. Int'l Conf. Peer-to-Peer Computing,* Sept. 2003.

[5] Y.-H. Chu, S.G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture," *Proc. ACM SIGCOMM,* pp. 55-67, 2001.

[6] Y.-H. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS,* pp. 1-12, June 2000.

[7] Y. Cui, B. Li, and K. Nahrstedt, "Asynchronous Streaming Multicast in Application-Layer Overlay Networks," *IEEE J. Selected Areas in Comm.,* special issue on recent advances in overlay networks, vol. 22, no. 1, pp. 91-106, Jan. 2004.

[8] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," *Proc. ACM Multimedia,* pp. 15-23, Oct. 1994.

[9] C. Diot, B.N. Levine, B. lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network Megazine,* special issue on multicasting, vol. 14, pp. 78-88, 2000.

[10] T.T. Do, K.A. Hua, and M. Tantaoui, "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment," *Proc. IEEE Int'l Conf. Comm. (ICC),* June 2004.

[11] D.L. Eager, M.K. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 5, pp. 742-757, Sept./Oct. 2001.

[12] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust Incentive Techniques for Peer-to-Peer Networks," *Proc. ACM Conf. Electronic Commerce (EC '04),* May 2004.

[13] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz, "Tune to Lambda Patching," *ACM Performance Evaluation Rev.,* vol. 27, no. 4, pp. 20-26, Mar. 2000.

[14] M. Guo and M.H. Ammar, "Scalable Live Video Streaming to Cooperative Clients Using Time Shifting and Video Patching," *Proc. IEEE INFOCOM,* 2004.

[15] B. Horne, B. Pinkas, and T. Sander, "Escrow Services and Incentives in Peer-to-Peer Networks," *Proc. Third ACM Conf. Electronic Commerce,* 2001.

[16] K.A. Hua, Y. Cai, and S. Sheu, "Exploiting Client Bandwidth for More Efficient Video Broadcast," *Proc. Int'l Conf. Computer Comm. and Networks,* pp. 848-856, Oct. 1998.

[17] K.A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," *Proc. ACM Multimedia,* pp. 191-200, Sept. 1998.

[18] K.A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *Proc. ACM SIGCOMM,* Sept. 1997.

[19] M. Kwon and S. Fahmy, "Topology-Aware Overlay Networks for Group Communication," *Proc. ACM Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* pp. 127-136, 2002.

[20] M. Kwon and S. Fahmy, "Path-Aware Overlay Multicast," *Computer Networks,* vol. 47, pp. 23-45, 2005.

[21] K. Lai, M. Feldman, I. Stoica, and J. Chuang, "Incentives for Cooperation in Peer-to-Peer Networks," *Proc. Workshop Economics of Peer-to-Peer Systems,* 2003.

[22] B.B.S. Banerjee and C. Kommareddy, "Scalable Application Layer Multicast," *Proc. ACM SIGCOMM,* pp. 205-217, 2002.

[23] S. Sheu, K.A. Hua, and W. Tavanapong, "Chaining: A Generalized Batching Technique for Video-on-Demand," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems,* pp. 110-117, June 1997.

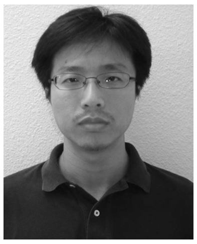
[24] D.A. Tran, K.A. Hua, and T.T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," *Proc. IEEE INFOCOM,* 2003.

[25] P.C.V. Padmanabhan, H. Wang, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," *Proc. ACM/IEEE Int'l Workshop Network and Operating Systems Support for Digital Audio and Video,* pp. 177-186, 2002.

[26] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting," *Multimedia Systems,* vol. 4, no. 4, pp. 179-208, Aug. 1996.

[27] X. Xu, Y. Wang, S.P. Panwar, and K.W. Ross, "A Peer-to-Peer Video-on-Demand System Using Multiple Description Coding and Server Diversity," *Proc. IEEE Int'l Conf. Image Processing (ICIP),* Oct. 2004.

Ying Cai received the PhD degree in computer science from the University of Central Florida in 2002. While studying at this university, Dr. Cai was the chief architect at nStor/StorLogic leading the effort to develop network storage technology. He developed the first remote and centralized RAID management system back in 1996 and the product was licensed by several major companies, including SGI, Adaptec, and NEC. Currently, Dr. Cai is an assistant professor in the Department of Computer Science at Iowa State University. His research interests include wireless networks, mobile computing, and multimedia systems. He is a member of the IEEE and the IEEE Computer Society.



Jianming Zhou received the BS degree in computer science from Shanghai Jiao Tong University, People's Republic of China, in 1999 and the MS degree in computer science from Iowa State University in 2006. He is currently a PhD student in the Computer Science Department at Iowa State University. His research interests include peer-to-peer networks and wireless sensor networks.