Towards Stadium-Scale Wireless Media Streaming

Jean Cao Carey Williamson
Department of Computer Science, University of Calgary
Email: {caox,carey}@cpsc.ucalgary.ca

Abstract

This paper proposes a scalable architecture for multimedia streaming in wireless LANs. Current IEEE 802.11 WLANs can support tens of media streaming users. We propose a hierarchical approach that can support over 1000 concurrent users. Our architecture exploits several existing technologies, including multi-channel WLANs, power control, and caching. The paper first describes the system design and operation, as well as assumptions and constraints. Performance issues are then explored via simulation, using synthetically-generated media streaming workloads. The simulation results show that: 1) our design can support up to 1600 concurrent media streaming clients using current WLAN technology; 2) the system architecture is quite robust to the user-level characteristics of the media streaming workload; and 3) proper cache management can make the system operate effectively even with limited cache sizes.

1 Introduction

IEEE 802.11 [10] wireless local area networks (WLANs) are currently in widespread use. At the same time, multimedia network applications, such as media streaming and video conferencing, are growing in popularity on the Internet.

802.11 WLANs can adequately support the QoS requirements of multimedia applications, as long as the number of users is limited. For example, Cao *et al.* [4] show experimentally that an 11 Mbps 802.11b ad hoc network can deliver good quality multimedia streams (400 kbps video and 128 kbps audio per user) for up to 8 clients. Adding one more client, however, overloads the system, degrading the performance for all users.

This paper studies wireless media streaming in a larger-scale WLAN. As an example scenario, consider a stadium, gymnasium, or arena setting where hundreds or thousands of spectators could use wireless networking technologies to access rich multimedia content during a live sporting event. If a WLAN blankets the stadium, then the spectators can use their personal wireless devices to view (on demand) an assortment of media objects, including replays, highlights, player interviews, advertisements, and live streaming feeds from different camera angles throughout the stadium.

Using a wireless infrastructure, rather than wired, offers the advantages of portability and easy deployment. However, the performance challenges are many, with scalability being the central issue. Traditional 802.11 WLANs cannot support hundreds of streaming users, for several reasons. First, 802.11 WLANs have limited bandwidth (11 Mbps for 802.11b, and 54 Mbps for 802.11a/g), and only about 60% of this capacity is effectively usable for end-to-end throughput [16]. Second, a wireless channel is shared by all users in the network. With hundreds of users, the per-user bandwidth share is inadequate for multimedia streaming.

Our work focuses on improving the scalability of wireless media streaming in 802.11 WLANs. In the *wired* Internet, many techniques have been proposed to make multimedia streaming scalable. These approaches include multicast [2, 8, 9, 20], proxy caching [19, 22], and Content Distribution Networks (CDNs) [1], to name a few. However, for multimedia streaming in 802.11 WLANs, none of these provide a complete solution on their own.

In this paper, we propose a scalable approach that combines ideas from *multi-channel* WLANs, *transmission power control*, and *cache-and-relay* technologies. The paper first describes the basic system design and operation, as well as the assumptions underlying our work. Performance issues are then explored via simulation, using different assumptions about the media streaming workload characteristics.

The rest of the paper is organized as follows. Section 2 discusses related work and technologies. Section 3 describes system architecture. Section 4 presents our simulation methodology. Section 5 presents simulation results. Section 6 concludes this paper.

2 Background and Related Work

Many techniques have been proposed to make media streaming scalable on the Internet. We discuss proxy caching, multicast, and coding as examples.

Liu *et al.* [19] evaluated four classes of caching strategies for media streaming, while Park *et al.* [22] proposed a two-layer proxy server architecture to increase streaming scalability. However, both studies focus on wired networks. In a WLAN, proxy servers interfere with each other because of the shared bandwidth. Thus proxy caching cannot be directly used in WLANs.

Multicasting is scalable. Moreover, it can benefit from the broadcast nature of wireless transmission. Many users requesting the same media object can join a multicast session by simply listening to the channel. However, multicast may not scale as expected when user interactions with media streams (e.g., rewind and fast forward) are involved [14]. Also, the server bandwidth cost of a multicast session might limit the number of sessions (for different media objects) co-existing in a WLAN (refer to Figure 2 of [1]).

Jenkac *et al.* [11] proposed a broadcasting approach that applies fountain codes to Harmonic Broadcasting (HB) [5]. It is scalable, and can provide asynchronous and reliable media streaming service in wireless environments. However, there is a sharp tradeoff between low start-up delay and high server bandwidth cost in HB. The processing cost for decoding on resource-poor mobile devices is also an issue. Moreover, as with most periodical broadcasting schemes, it cannot support user interactions well.

To address the aforementioned limitations, we propose a new scalable approach for wireless media streaming. In the following, we review three key technologies used in our solution: multi-channel WLANs, transmission power control, and caching-and-relay technologies:

Multi-channel: The term *multi-channel* refers to wireless technology that can use more than one radio channel. For example, Mishra *et al.* [21] demonstrate that 802.11b/g networks can support up to 4 radio channels concurrently. Some wireless devices achieve this property using multi-radio systems, with each interface communicating on a different physical channel. Other devices have just a single radio transceiver, which is tunable to any of the available channels.

Multi-channel systems are commercially available now for infrastructure-based WLANs [6]. Using multiple channels increases the number of WLAN users that can be supported, while improving mobility management and traffic load balancing [18].

Multi-channel systems are also valuable in multi-hop wireless ad hoc networks. The use of multiple physical-layer channels reduces contention among neighbouring nodes, increasing concurrency in frame transmissions and improving throughput for the network [17].

Power Control: Power control is a well-known technique to limit the signal coverage area by reducing the transmission power [7]. With power control and/or directional antennas, mobile users can be geographically divided into smaller groups. In each group, contention and collisions are significantly reduced. However, merely limiting the number of users per group does not guarantee sufficient bandwidth for streaming traffic, since the available bandwidth is still shared with incoming and outgoing traffic. This problem can be alleviated using multi-channel technology.

Cache-and-Relay: The idea of cache-and-relay extends caching technology for media streaming [3, 13, 15]. A streaming client caches a stream while displaying the media and, if needed, can relay the stream to other downlink clients that later request the same stream. This approach can save significant bandwidth. With cache-and-relay, all the streaming clients in a network are managed in a tree, with the origin server as the root, relay nodes as the intermediate branching points, and end users as leaf nodes.

A cache-and-relay system resembles a multicast tree, but there are two main differences. First, multicast does not typically provide caching (though some techniques use prefix caching to reduce startup delays). Second, a multicast session provides one stream object only. In cache-and-relay, a relay node can provide content for any streams that it has cached.

3 System Architecture

In this section, we describe a hierarchical approach for wireless media streaming. Our system architecture leverages ideas from multi-channel WLANs, power control, and cache-and-relay technologies. The goal is to support hundreds or thousands of wireless media streaming users in a stadium WLAN. Though we focus on unicast in this paper, our system can easily be extended to support multicast, with greater scalability.

3.1 Network Setup

Figure 1 illustrates our proposed architecture for scalable wireless media streaming. Figure 1(a) provides a conceptual (side) view of the system, illustrating its hierarchical structure. Figure 1(b) provides a bird's eye (top) view of the system, illustrating its spatial structure. Further description of the system follows.

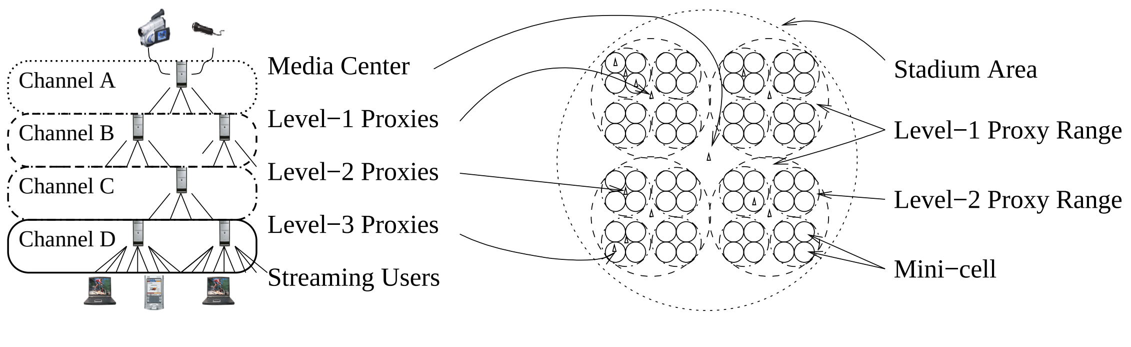


Figure 1. System Architecture for Scalable Wireless Media Streaming

The topmost component in Figure 1(a) is the *media center*, which is the central repository for all media objects generated, stored, and accessed in the stadium WLAN. The media center includes a streaming server, referred to as the *origin* server, as well as ample storage space for dozens of media objects. For ease of exposition, our discussion assumes *stored* (i.e., pre-recorded) media objects, though the media center can certainly have facilities (e.g., video camera, microphone) for live streams as well. For simplicity, we assume Constant Bit Rate (CBR) media streams.

With a single wireless transceiver at the media center, only a limited number of users could be accommodated. We express the "capacity" in terms of the number N of simultaneous CBR streams that can be supported. The value of N depends on the WLAN technology chosen (e.g., 802.11b or 802.11g) and the media quality provided (e.g., 128 kbps or 500 kbps per stream). The default value $N = 32$ models an 802.11g WLAN with 500 kbps per stream.

To increase the total number of streaming users supported, cache-and-relay nodes are used between the media center and the streaming users. As shown in Figure 1, these nodes are organized in a tree structure with the media center as the root. A system parameter k specifies the number of levels in the hierarchy. Media requests from users propagate up the hierarchy, while media streams are delivered down the hierarchy.

Each level beneath the media center contains a set of cache-and-relay nodes. We refer to these nodes generically as proxies[1]. The proxies perform on-demand caching for media objects, on a block basis, where a *block* is a fixed-size or fixed-duration (e.g., 1 second) segment of a media object.

[1]The proxies could be special nodes deployed in the WLAN by the network designer, or they could be regular clients in the ad hoc network that are well-resourced, spatially well-positioned, and willing to function as a proxy.

The media center communicates directly with the Level-1 proxies, which in turn communicate directly with the Level-2 proxies, and so on. In Figure 1, $k = 3$, so the Level-3 proxies communicate directly with the wireless streaming clients.

Wireless channel usage is carefully coordinated in our system. Because all Level-1 proxies must be within transmission range of the media center, frame transmissions by these proxies could possibly interfere with each other. To avoid this interference problem, we use power control to constrain the downstream transmission range of each proxy, as shown in Figure 1(b). Circular regions of coverage are assumed, with the radius decreasing as you descend the hierarchy. Each level of the hierarchy operates on a different wireless channel[2], as indicated in Figure 1(a).

This wireless layout minimizes intra-level and inter-level interference for the proxies. As a result, the lowest-level "mini-cells" in our architecture can each support up to N concurrent media streams. Increasing the number of levels in the hierarchy, or the branching factor at each level, increases the overall number of concurrent media streams that can be supported.

For maximum system capacity, each proxy node needs *two* wireless network interfaces, with each operating on a different channel. One interface is used for parent communication (i.e., propagating user requests upstream, and receiving incoming media flows to send downstream), while the other is used for child communication (i.e., receiving incoming user requests, and sending outgoing media flows downstream). This is the default system setup in our study.

This setup provides concurrency between incoming and outgoing media streams. That is, even though each wireless interface is half-duplex, the two interfaces op-

[2]For 802.11b/g, there can be up to 4 different channels concurrently in use. The 3-level hierarchy in our example respects this constraint.

erate independently. If only a single interface is available, the proxy itself operates as "half-duplex", and system throughput is halved.

3.2 System Operation

Admission control is important for stable operation of the system. The maximum number of outgoing streams that can be supported at any level is $N = 32$.

When a user generates a request at the leaf level, the request is sent to the proxy for that user's mini-cell. Whether the request can be served or not depends on the request type (e.g., start, stop, rewind, fast forward), and the current resource availability in the WLAN.

```
If the proxy (or server) node has no capacity
Then block the request
Else If the node has the desired content
     Then
          Grant the request.
          Start a new stream from this node.
          Update resource consumption for this
          node and all lower-level proxies
          along the path to the client.
     Else
          Recursively check the parent proxy
          (or origin server) with this procedure.
```

Figure 2. Admission Control Algorithm

In our system model, an incoming request to start a new media stream is handled as shown in Figure 2. Note that the media center stores *all* of the media objects in the system, so the content can always be found. Requests at the root level are rejected only when channel capacity is fully committed.

If proxies have infinite-size caches, then the initial requests for media objects replicate content along paths of the tree, and subsequent requests for the same objects are served from caches close to the clients. However, with finite cache sizes, cache misses will occur.

Cache misses sometimes lead to the blocking of requests. Satisfying a miss requires media movement across at least one level of the caching hierarchy, which may or may not be possible, depending on the current resource usage. Blocking can happen at any level of the hierarchy, at any time.

User interactions with the media stream (e.g., fast forward, rewind) can exacerbate cache misses. In fact, it is possible for an in-progress media streaming session to be interrupted (terminated). For example, consider a session that is currently streaming from a Level-3 cache, when a user fast forwards to a new portion of the object that is not yet in the cache. Acquiring the missing pieces of the media object from a higher-level cache (or the origin server) may not be possible if upstream resources are fully in use. The media streaming session is terminated. This unfortunate outcome is called a dropped session.

Cache management plays an important role in our system. The main issues are the cache size and the cache replacement policy (see Section 5.2).

3.3 System Capacity

Our system architecture can be modelled as a tree. Hence, we can analyze the system scaling characteristics using its tree properties. Starting from the root, the branching factor from the media center to the Level-1 proxies is L_1. In general, the branching factor to the Level-i proxies is L_i, where $L_i \leq N$. The L_i values can be set independently at each level.

The maximum number N_{max} of users that can be supported depends on N and on the branching factor at each of the k levels of proxies:

$$N_{max} = N \prod_{i=1}^{k} Li \tag{1}$$

Figure 1 shows an example of a 3-level hierarchy with 4 cache-and-relay proxies in each level. For this system, $N_{max} = 2048$. This calculation assumes that each user has one active unicast media stream.

Careful system design needs to consider the number of levels as well as the number of proxies in each level. Clearly, increasing the branching factor increases N_{max}. However, the L_i values cannot be increased arbitrarily, because wireless transmission range cannot be precisely controlled. Also, increasing the branching factor constrains the number of media streams that can be allocated on each downstream link.

3.4 Capacity Constraints

Whether the system can actually support N_{max} users or not depends on caching performance (i.e., hit/miss rates) and system dynamics (i.e., timing and location of user requests, and the media objects desired). With cache-and-relay, each proxy can cache media content that it has received in the past, keeping as much useful content as possible (depending on its cache size and replacement policy). With infinite cache size, the problem is trivial: all the media content is eventually cached in the proxies closest to the users. As long as users are evenly distributed across the mini-cells, N_{max} can be achieved.

With finite caches, and perhaps new content generated at the media center, cache misses are more frequent. As mentioned earlier, cache misses can lead to request blocking and/or session dropping. The worst case is if each user requests a distinct media object. In this extreme situation, the root level of the hierarchy is the bottleneck: system capacity is limited to at most N concurrent media streams.

In summary, the proposed system can achieve N_{max} only if the following two constraints are satisfied. First, each mini-cell must have N users. Second, the cache hit rate at each proxy must be high enough to limit its upstream workload demands to at most N/L_i media streams (on average) from its parent. We call this constraint the *filtering bound*.

These constraints, particularly the filtering bound, have several important implications:

- User interactions such as "rewind" and "fast forward" can increase the cache miss rate, making N_{max} unachievable. See Section 5.1.
- Cache size and cache replacement policy affect the cache hit rate, and thus the system scalability. We explore this issue in Section 5.2.
- The *filtering bound* depends on L_i, the branching factor at Level-i. Thus, system parameters might affect achievable system performance. We compare different architectures in Section 5.3.
- As the size and diversity of the media object collection increases, the cache miss rate may suffer. We study this effect in Section 5.4.

4 Simulation Methodology

4.1 Overview

We use simulation to study the scalability characteristics of the proposed system. There are four main factors of interest in our study: the effects of media user interactions, caching issues, system parameters, and the size of the media streaming workload.

The system architecture is defined by specifying the number of levels k and the branching factor L_i at each level. For example, the architecture shown in Figure 1 uses $k = 3$ and $L_1 = L_2 = L_3 = 4$. This example is the default configuration for most of our experiments. For convenience, we refer to this design as "4.4.4".

The primary performance metric in our study is the number of simultaneous user media streams that can be supported. Secondary performance metrics include the number of blocked requests, the number of dropped media streaming sessions, and the average number of outgoing media streams at each level of the hierarchy.

4.2 Simulation Assumptions

The following assumptions simplify our study:

– Users do not move. A streaming session, once started, remains in the same mini-cell until it terminates or is dropped.

– Existing solutions for caching, relaying, and power control can be used to deploy our system.

– Cache-and-relay nodes have two 802.11g interfaces, so that $N = 32$.

– The hierarchy is homogeneous and balanced. That is, each level uses the same branching factor, and each mini-cell has a similar number of users.

– The downstream capacity for media content delivery is the bottleneck. Bandwidth consumption for sending user requests upstream is ignored.

4.3 Workloads

The media-streaming workloads for our simulator are generated using GISMO (Generator for Internet Streaming Media Objects [12]). GISMO provides control over the number, size, and popularity of media objects, as well as the user-level access characteristics. GISMO generates workloads at the session level and request level. A session corresponds to a single user generating 1 or more requests for 1 media object.

In our initial experiments (Sections 5.1, 5.2, and 5.3), we use workloads that are as simple as possible, to better understand system behaviour. These workloads have a *single* read-only media object that is 7200 seconds (2 hours) long. The session arrival process is Poisson, with an average rate of 1000 session arrivals per hour. Each session starts playback at the beginning of the media object.

The workloads differ in how the users interact with the media objects. There are four different models in our study. The first user model, and the simplest, is the *Passive* user. Such a user views a media object in its entirety, from start to finish, with no additional interactions. The second user model is *Early Termination.* Such a user views a media object in a sequential fashion, but may issue a "stop" command at any time. The third user model includes *Rewind* functionality. Such a user may jump backwards in the media stream at any time, to repeat viewing an earlier portion of the object. The fourth user model is called *Random Jump.* Such a user may rewind or fast forward (skip) within the media object at any time. Random Jump provides the most complete model of user-level behaviour.

Statistical distributions are used to model the user characteristics. In the Early Termination model, the playback duration is modelled using a bounded Pareto distribution. For Rewind and Random Jump users, playback can jump to a different location in the media object, from which sequential playback occurs until the next jump or the 2-hour play time is reached. The jump distances are modelled using Pareto. The time between jumps are modelled using an exponential distribution, with a parameter λ specifying the frequency of jumps [12]. We use the default GISMO parameters for these models.

In Section 5.4, we use a more complicated media streaming workload with 32 media objects. Object popularity follows a Zipf-like distribution, and object sizes follow a Lognormal distribution, both using default parameters in GISMO. Early Termination and Random Jump are both considered.

4.4 Simulator Overview

We developed our own event-driven simulator of the proposed system. Input parameters specify the system architecture (e.g., 4.4.4). A media streaming workload file is provided as input. The simulator operates at the media stream level; the network protocol stack is not simulated. For each new user request in the workload, the user is mapped to a mini-cell uniformly at random. The proxy in that mini-cell starts the procedure in Figure 2 to determine system resource avalability. If the request is granted, the streaming session starts and continues until either the session finishes or a jump happens. Upon a jump, the admission procedure is checked again. An existing streaming session can be dropped if resources are unavailable.

5 Simulation Results

This section presents the results from our simulation experiments. The experiments study the sensitivity to certain workload factors and system design parameters.

5.1 User-Level Workload Behaviour

The first experiment studies the effect of user-level behaviour on the system operation. All tests in this section use a 4.4.4 system, for which $N_{max} = 2048$. All proxy nodes have infinite cache size.

Our initial validation tests (not shown here) used light traffic loads. The results show correct system operation, with no blocking or dropping observed. All requests are granted, and complete successfully. The following tests study system behaviour under overload.

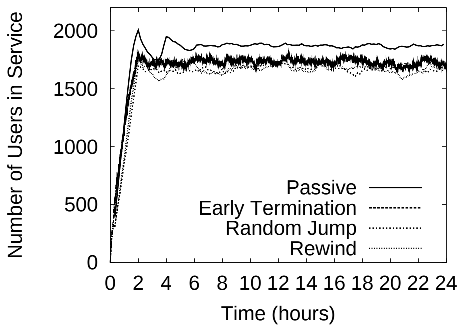


Figure 3. Number of concurrent media streams for different user behaviour models

For the Passive user model, we study system saturation when handling 1440 session arrivals per hour. This arrival rate exceeds the system capacity (1024 requests per hour) determined from Little's Law (since each request is for a 2-hour media object).

Figure 3 presents the simulation results. The graph shows the number of simultaneous user streams in service, as a function of simulation time. There are four lines on the graph, one for each user-level workload model considered.

Figure 3 shows that the system reaches steady-state within a few hours, and remains stable for the rest of the simulation duration (24 hours). The Passive workload model saturates with about 1800 sessions in service at a time. This number is lower than N_{max} because of system dynamics (i.e., unbalanced load across mini-cells), and the request blocking that occurs. No session dropping is possible for the Passive user model.

We next consider the Early Termination model, in which subsequent requests for the same media object can sometimes lead to session dropping. Since the average session duration (2448 seconds) is shorter than for the Passive user model (7200 seconds), we saturate the system by sending 3000 session arrivals per hour. The simulation results in Figure 3 show that the system can sustain about 1700 concurrent streams.

For Rewind and Random Jump users, session dropping is even more likely. The workload consists of 1000 session arrivals per hour. Figure 3 shows that for these two user models, around 1600 concurrent streams can be supported.

User interaction does affect the system steady-state behaviour. However, these impacts are not dramatic.

5.2 Caching Issues

Caching plays a key role in our system. The previous results in Section 5.1 look promising, but those results assumed infinite cache size, which is not realistic.

In this section, we study the issue of cache size and cache replacement policy. These simulations assume a 4.4.4 system, the Random Jump workload model, and 1000 requests per hour.

We consider three cache replacement policies, namely Random (a simple baseline policy), Least-Recently-Used (LRU), and Least-Frequently-Used (LFU)[3]. We vary cache sizes from 0 (none) to 7200 blocks (infinite), using intermediate values of 100, 200, 400, 800, 1600, 3200, and 6400 blocks.

Selected simulation results are presented in Figure 4. Figure 4(a) shows the results for the Random cache replacement policy, while Figure 4(b) shows the results for LRU, and Figure 4(c) shows the results for LFU.

There are two main observations from these results:

- When there is no cache, only $N = 32$ users can be accommodated at a time (note the line just above the horizontal axis). This result is obvious, since the media center becomes the bottleneck.

- With an adequate cache size, all caching policies can support about 1600 concurrent media users. However, the caching policies differ in their effectiveness at smaller cache sizes. LFU in Figure 4(c) is the most effective policy; the system reaches the same steady-state level as long as the cache size is at least 100 blocks (1.4% of the media object size). Random and LRU are less effective. Random in Figure 4(a) needs a cache size of 800 blocks (11% of object size) to reach its steady-state level, while LRU in Figure 4(b) needs at least 1600 blocks (22% of object size).

  LFU outperforms LRU and Random because of the characteristics of the media streaming traffic models. GISMO workloads are biased toward the beginning portion of media objects. LFU exploits this property well, and tends to retain the popular initial portion of the media object in cache.

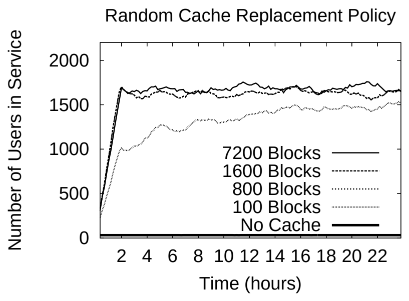
These results show that with caching, the system can support 1600 users. Furthermore, this can be achieved with modest cache sizes.

We further study the dynamic evolution of our system over time, to gain insights into the effects of cache size and cache management. Figure 5 shows how cache size and replacement policy affect the system resource usage at each level. Each graph shows a time series plot of the (average) number of concurrent downstream media streams at each level of the hierarchy. The rows of graphs represent different cache sizes, from 0 to 1600 blocks. From left to right, the graphs correspond to Random, LRU, and LFU replacement, respectively.

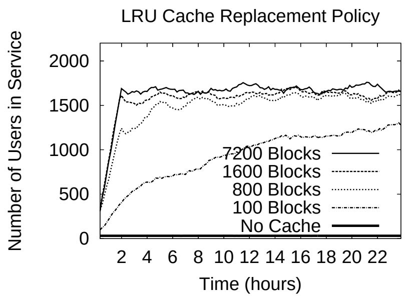
When no cache is present, the bottleneck is at the root of the hierarchy. The media center quickly saturates with $N = 32$ concurrent media streams, and remains there throughout the simulation. The 64 Level-3 proxies each have 0.5 media streams, on average.

As the cache size is increased in subsequent rows of Figure 5, different transitions occur for the caching policies. For example, the LFU policy functions well with a cache size of 100 blocks. With this cache size,

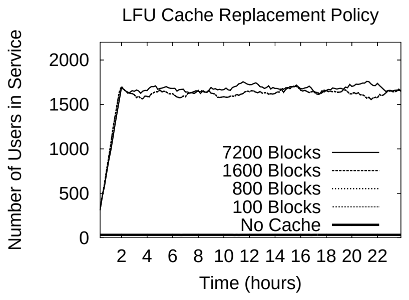
[3]In reality, LFU with aging can be applied to avoid the "cache pollution" problem.



(a) Random

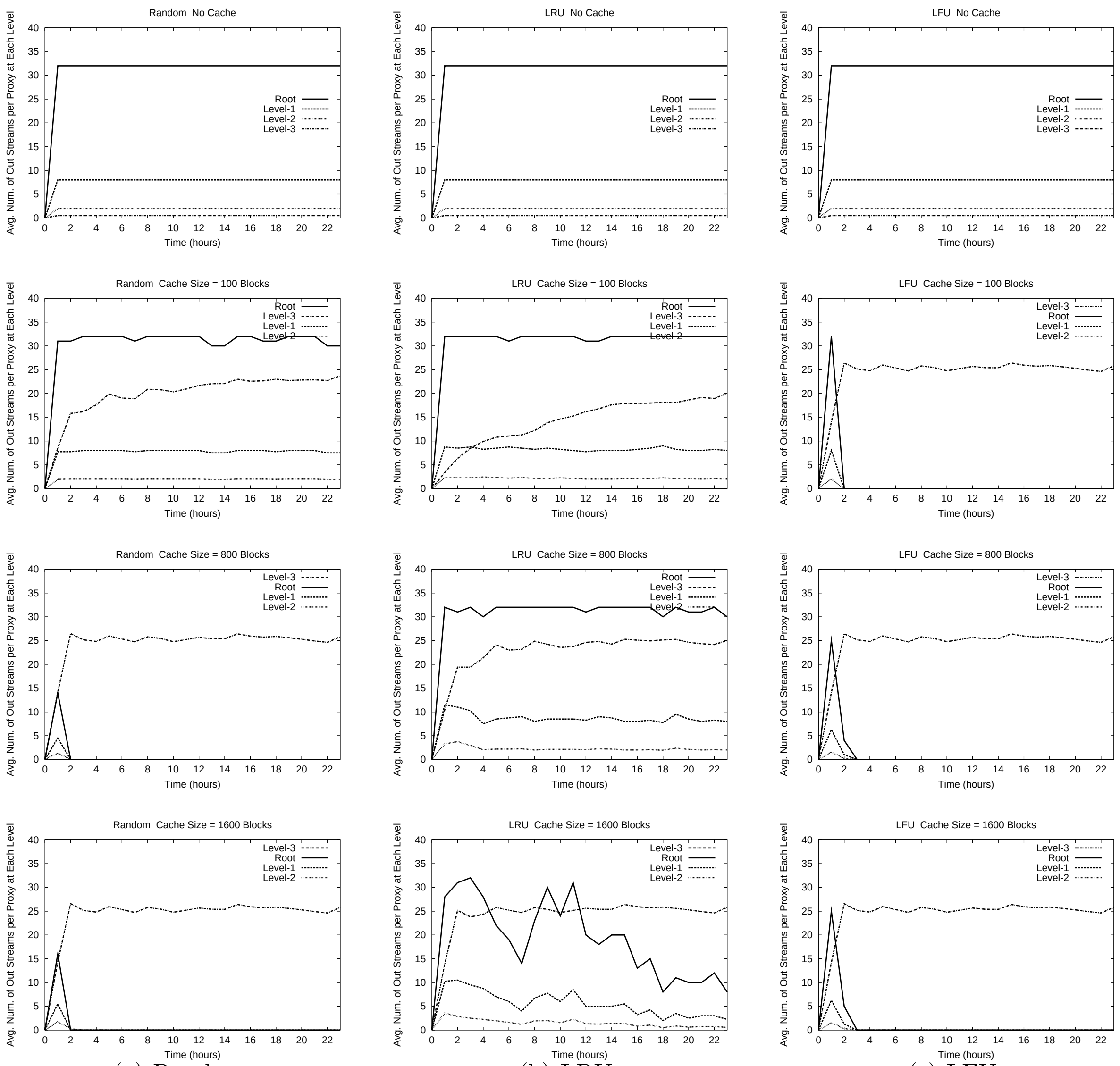


(b) LRU



(c) LFU

Figure 4. Comparison of different caching replacement policies



(a) Random (b) LRU (c) LFU

Figure 5. System evolution for different cache sizes and replacement policies

the useful media content is pulled down the hierarchy toward the clients, offloading the higher levels of the hierarchy. The system reaches steady-state with an average of 25 concurrent user media streams at each leaf-level proxy. For the same cache size of 100 blocks, Random and LRU remain saturated at the root level.

When the cache size is 800 blocks, the Random policy achieves the same steady-state behaviour as LFU. Since there are multiple levels of cache (each 800 blocks in size and managed independently), many media object requests can be satisfied within the hierarchy. The root level is no longer the bottleneck.

At a cache size of 1600 blocks, the LRU policy also transitions toward the same steady-state behaviour as LFU, albeit rather slowly. There are three reasons for the poor performance of LRU. First, GISMO workloads bias requests to the initial portion of media objects. LFU exploits this characteristic better than LRU does. Second, the blocks of the media objects tend to be accessed sequentially when media objects are viewed. Sequential reference patterns do not work well with LRU caches when the object size exceeds the cache size, and recency becomes a poor predictor. Third, the multiple levels of caches all have the *same* cache size and the *same* replacement policy. If one level of cache has evicted some needed blocks of the media object, then it is likely that the next higher level of cache has done so as well. These characteristics make the LRU policy a poor choice (even worse than Random) for our media streaming workload scenario.

The simulation results in Figure 5 illustrate how our system achieves its scalability. These results also show why LFU has the best performance in Figure 4.

5.3 System Architectural Parameters

As stated in Section 3.4, the system architecture determines the maximum number of media streams that can be supported. The system parameters specify the number of levels and the branching factor at each level. Stable system operation is dependent upon the cache filtering bound at each level, and the branching factors.

In this section, we compare four different configurations of our system, all with the same $N_{max} = 2048$. The configurations are 4.4.4, 2.4.8, 8.4.2, and 8.8. A system with a larger branching factor L_1 at the top has tighter constraints on the number of media streams deliverable to each Level-1 proxy.

The purpose of the experiment is to understand the effects of architectural parameters on system operation, particularly when the cache size is small. These simulation experiments use the Random Jump workload model, with 1000 session arrivals per hour for a single 2-hour media object. All three cache replacement policies (Random, LRU, and LFU) were tested.

For both Random and LFU, the system reaches its steady-state saturation even with small cache sizes, so the influence of the architecture is negligible.

For LRU cache replacement, the architecture does have some influence on the results, but only when the cache size is small (i.e., 100 blocks). When the cache size is small, the bottleneck remains at the root. The 2.4.8 design offers slightly better performance than 4.4.4, because of its filtering bound advantages. The 8.4.2 and 8.8 architectures perform slightly worse than 4.4.4. (For space reasons, the graphs are not shown.)

In general, when the cache size is large enough so that the root is not the bottleneck, the branching factor has minor influence on the system scalability. There might be minor differences if small LRU caches are used, but LRU caches are *not* recommended for the workload characteristics observed in our simulations.

5.4 Multiple Media Objects

In this section, we consider a more complex workload with 32 media objects. The purpose of the experiment is to understand the system performance for more realistic workloads. We use the Random Jump user model, and 1000 session arrivals per hour.

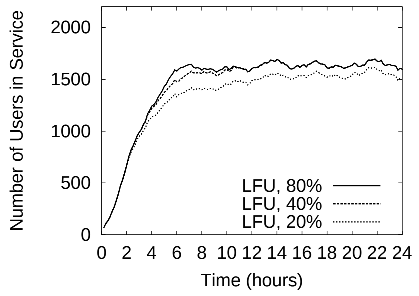


Figure 6. Simulation results for a larger workload with 32 media objects

Figure 6 shows the results from this simulation experiment. The graph shows the number of user media streams concurrently in service, as a function of simulation time.

The results from this experiment reinforce those observed in earlier experiments. In steady-state, the system can support over 1500 concurrent user media streams. Caching plays a key role in our architecture, and LFU cache replacement is best. LFU exploits not only the bias toward the start of each media object, but also the skewed object popularities. Caching is effective, even at modest cache sizes. The results in Figure 6 consider cache sizes ranging from 20% to 80% of the cumulative size of the media objects. A cache size of 20% is adequate to attain reasonable system performance. There is little benefit as the cache size is increased further. Once the system is in steady-state, few active streaming sessions are dropped.

6 Conclusions and Future Work

This paper proposes a scalable architecture for wireless media streaming in stadium-scale WLANs. Simulation experiments show that our approach can support up to 1600 concurrent user media streams, using existing features of 802.11 technology. Our hierarchical approach leverages multi-channel WLANs, transmission power control, and cache-and-relay technology to build a scalable wireless media streaming network.

Caching plays a central role in our system design. For the media streaming workloads considered in our work, LFU is the most effective cache replacement policy to use. This policy ensures scalable system operation, even with modest cache sizes.

The simulation results show that our approach is promising. However, much additional work remains.

One major task is to assess the practicality of our system, by developing and experimenting with a proof-of-concept prototype. Experimental work will provide a greater understanding of the many wireless networking issues inherent in our approach. Experiments with user mobility and with a broader range of media streaming workloads are also required.

Acknowledgements

Financial support for this research was provided by iCORE, NSERC, and TRLabs. The authors thank the anonymous MASCOTS reviewers for their constructive comments on an earlier version of this paper.

References

[1] J. Almeida, D. Eager, M. Ferris, and M. Vernon, "Provisioning Content Distribution Networks for Streaming Media", *Proceeding of IEEE INFOCOM Conference*, New York, NY, pp. 1746-1755, June 2002.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast", *Proceeding of ACM SIGCOMM Conference*, Pittsburgh, PA, pp. 205-217, August 2002.

[3] A. Bestavros and S. Jin, "OSMOSIS: Scalable Delivery of Real-Time Streaming Media in Ad-Hoc Overlay Networks", *Proceedings of IEEE ICDCS Workshop on Data Distribution in Real-Time Systems*, Providence, RI, pp. 184-195, May 2004.

[4] X. Cao, G. Bai, and C. Williamson, "Media Streaming Performance in a Portable Wireless Classroom Network", *Proceedings of IASTED EuroIMSA*, Grindelwald, Switzerland, pp. 246-252, February 2005.

[5] L. Engebretsen and M. Sudan, "Harmonic Broadcasting is Bandwidth-Optimal Assuming Constant Bit Rate", *Proceeding of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, January 2002.

[6] Engim, Inc., "EN-3001 Intelligent Wideband WLAN Technology", http://www.engim.com/products_technology.html.

[7] J. Gomez-Castellanos and A. Campbell, "A Case for Variable-Range Transmission Power Control in Wireless Multihop Networks", *Proceedings of IEEE INFOCOM*, Hong Kong, pp. 1425-1436, March 2004.

[8] A. Hu, "Video-On-Demand Broadcasting Protocols: A Comprehensive Study", *Proceedings of IEEE INFOCOM*, Anchorage, AK, pp. 508-517, April 2001.

[9] K. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-On-Demand Systems", *Proceedings of ACM SIGCOMM Conference*, Cannes, France, pp. 88-100, September 1997.

[10] IEEE, "Wireless LAN Medium Access Control and Physical Layer Specifications", 1999.

[11] H. Jenkac, T. Stockhammer, and W. Xu, "Asynchronous and Reliable On-Demand Media Broadcast", *IEEE Network*, Vol. 20, No. 2, pp. 14-20, March 2006.

[12] S. Jin and A. Bestavros, "GISMO: A Generator of Internet Streaming Media Objects and Workloads", *ACM Performance Evaluation Review*, Vol. 29, No. 3, pp. 2-10, December 2001.

[13] S. Jin and A. Bestavros, "Cache-and-Relay Streaming Media Delivery for Asynchronous Clients", *Proceedings of the 4th International Workshop on Networked Group Communication*, Boston, MA, October 2002.

[14] S. Jin and A. Bestavros, "Scalability of Multicast Delivery for Non-Sequential Streaming Access", *Proceedings of ACM SIGMETRICS*, Marina del Rey, CA, pp. 97-107, June 2002.

[15] S. Jin, A. Bestavros, and A. Iyengar, "Accelerating Internet Streaming Media Delivery Using Network-Aware Partial Caching", *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, pp. 153-160, July 2002.

[16] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical Maximum Throughput of IEEE 802.11 and its Applications", *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, pp. 249-256, April 2003.

[17] T. Kuang and C. Williamson, "A Bidirectional Multichannel MAC Protocol for Improving TCP Performance on Multihop Wireless Ad Hoc Networks", *Proceedings of ACM/IEEE MSWiM*, Venice, Italy, pp. 301-310, October 2004.

[18] T. Kuang, Q. Wu, and C. Williamson, "MRMC: A Multi-Rate Multi-Channel MAC Protocol for Multi-Radio Wireless LANs", *Proceedings of SCS SPECTS*, Philadelphia, PA, pp. 263-272, July 2005.

[19] J. Liu and J. Xu, "Proxy Caching for Media Streaming Over the Internet", *IEEE Communications*, Vol. 42, No. 8, pp. 88-94, August 2004.

[20] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel, "Scalable On-Demand Media Streaming with Packet Loss Recovery", *Proceedings of ACM SIGCOMM*, San Diego, CA, pp. 97-108, August 2001.

[21] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh, "Exploiting Partially Overlapping Channels in Wireless Networks: Turning a Peril into an Advantage", *Proceedings of ACM IMC*, Berkeley, CA, pp. 311-316, October 2005.

[22] Y. Park, K. Baek, and K. Chung, "Reducing Network Traffic Using Two-Layered Cache Servers for Continuous Media Data on the Internet", *Proc. of IEEE COMPSAC*, Taipei, Taiwan, pp. 389-394, October 2000.