AMuSe: Large-scale WiFi Video Distribution - Experimentation on the ORBIT Testbed

Varun Gupta†, Raphael Norwitz‡, Savvas Petridis‡, Craig Gutterman†, Gil Zussman†, Yigal Bejerano*
† Electrical Engineering, Columbia University, New York, NY, USA.
‡ Computer Science, Columbia University, New York, NY, USA.
* Bell Labs, Nokia, Murray Hill, NJ, USA.

***Abstract*—Currently, wireless video distribution cannot be provided in crowded venues due to resource limitations. In our recent papers we proposed *AMuSe*, a scalable system for WiFi multicast video delivery. The system includes a scheme for dynamic selection of a subset of the receivers as feedback nodes and a rate adaptation algorithm *MuDRA* that maximizes the channel utilization while meeting QoS requirements. We implemented *AMuSe* in the ORBIT testbed and evaluated its performance with 150-200 nodes. We present a dynamic web-based application that demonstrates the operation of *AMuSe* based on traces collected on the testbed in several experiments. The application allows to compare the performance of *AMuSe* with other multicast schemes and evaluate the performance of video delivery.**

I. Introduction

Multimedia (e.g., video) delivery is an essential service for wireless networks and several solutions were proposed for content delivery in crowded venues [1]. Most of them are based on dense deployments of Access Points (APs) which requires considerable expenditure and may suffer from interference between APs or hidden node problems. Some schemes [2] have demonstrated impressive ability to deliver video to a few nodes by utilizing Forward Error Correction (FEC) codes and retransmissions. However, most approaches do not scale to large groups with hundreds of nodes.

Multicast offers another approach for video delivery to large groups of users interested in venue specific content (e.g., sports arenas, entertainment centers, and lecture halls). However, WiFi networks provide *limited multicast support at a fixed low rate (e.g., 6Mbps for 802.11a/g) without a feedback mechanism to guarantee service quality*. To improve multicast performance, there is a need for a system that *dynamically adapts the transmission rate*.

We have been developing the Adaptive Multicast Services (*AMuSe*) system for content delivery over WiFi multicast to a large number of users. In [3], we focused on efficient feedback collection mechanisms for WiFi multicast as part of the *AMuSe* system. In [4], we presented a multicast rate adaptation algorithm (*MuDRA*). *MuDRA* leverages efficient multicast feedback collection for WiFi multicast of *AMuSe* and *dynamically adapts the multicast transmission rate to maximize channel utilization while meeting performance requirements*. Fig. 1 shows the overall *AMuSe* system composed of (i) *MuDRA* algorithm, and (ii) a feedback mechanism.

We implemented *MuDRA* with the *AMuSe* system on the ORBIT testbed, evaluated its performance with hundreds of

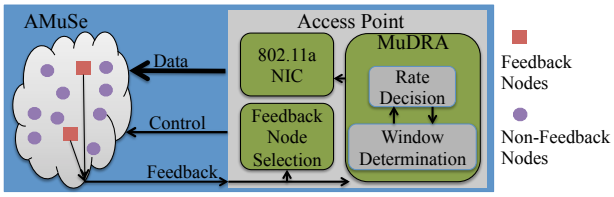

Fig. 1. The Adaptive Multicast Services (*AMuSe*) system consisting of the Multicast Dynamic Rate Adaptation (MuDRA) algorithm and a multicast feedback mechanism.

IEEE 802.11 nodes (between 150–200), and compared it to other multicast schemes. In this demonstration, we present an interactive web-based application that illustrates the performance of the overall *AMuSe* system based on experimental traces collected on the ORBIT testbed. We collected the traces over several days in different experimental settings with 150-200 nodes. Each experimental trace consisted of channel measurements at each node using several metrics such Link Quality, Packet Delivery Ratio (PDR) etc.

The application allows us to demonstrate different scenarios such as different channel conditions, interfering transmissions etc. For each scenario, the application shows the dynamic conditions over a period of time on the testbed from the appropriate experimental traces. An application can be used to compare the performance of several multicast schemes such as pseudo-multicast, unicast transmissions etc. We also demonstrate the performance of these schemes in different scenarios that have been measured on the testbed (e.g. interference, other WiFi flows, etc.) as well as syntactic scenarios based on manipulating the measured data. We note that the application is flexible and can be used for testing even more scenarios and algorithms in the future.

II. AMuSe Overview

AMuSe consists of two main components:

(i) **Feedback node selection**: The feedback scheme is based on the observation that adjacent nodes experience similar channel quality and interference patterns. *AMuSe* dynamically selects a small set of well-distributed nodes as *feedback (FB) nodes* according to the service quality that they experience. One approach for selecting FB nodes is dividing the nodes into a few clusters based on adjacency of nodes and maximum cluster size (D m). In each cluster, one or a few nodes with the weakest channel quality are selected as FB node. The FB

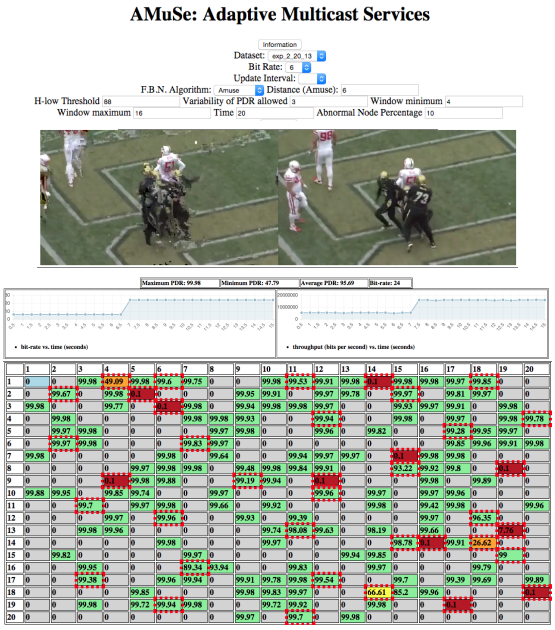

Fig. 2. A screenshot of the demo application. The control panel for selecting the feedback and *MuDRA* algorithm parameters is on the top. The video of two selected nodes is shown below. In this example we show one node with poor quality and one with good quality video. The multicast throughput and other metrics are in the graphs. The performance of the client nodes is shown on the grid where numbers in each box indicate the PDR and the color of the box indicates the range of PDR. The nodes highlighted with a red border are FB nodes and nodes in grey are non-functional due to hardware issues.

nodes periodically update the AP about their service quality, e.g., channel quality and PDR.

(ii) ***MuDRA*** : *MuDRA* is based on the observation that when the multicast rate exceeds an optimal rate, termed as target-rate, numerous receivers suffer from low PDR and their losses cannot be recovered. To overcome losses due to sampling, *MuDRA* includes a mechanism, called target condition, to detect when the system operates at the target-rate. *MuDRA* makes RA decisions based on the target condition and employs a dynamic window based mechanism to avoid rate changes due to bursts of interference.

III. Testbed Environment

The ORBIT testbed is a dynamically configurable grid of 400 nodes equipped 802.11 NICs. The separation between adjacent nodes is 1m. In our experiments, the node at one of the corners serves as a single multicast AP, configured in master mode. We use the 5GHz WiFi band for all experiments to minimize interference. We used the lowest supported transmission power of $1mW = 0dBm$ to compensate for the relatively small span of the testbed, which is only 28m. The other nodes are configured in managed mode and act as receivers. Throughout each experiment, the receivers collected traces of the channel quality, PDR values, and lists of correctly received packets. This information is later correlated with the transmission bit-rate records collected by the AP. We only choose nodes equipped with *Atheros 5212/5213* wireless cards with *ath5k* wireless driver for our experiments to avoid hardware/software mismatch.

IV. Demonstration Application

We developed an interactive web-based application to demonstrate the performance of *AMuSe*. The application has three main components: (i) the back-end where the experimental data is stored and managed, (ii) the front-end which provides the user interface, and (iii) a video tool for generating video streams. The front-end is web-based and can operate on any standard browser while the back-end requires installation of easily available open source libraries. For any experimental condition, the video tool generates the video stream received by a selected node. It maps video payload to UDP packets and discarding lost packet, according to the nodes traces.

The front end is built using Angular [5] which is a JavaScript framework for rendering dynamic features on web applications. The FB node selection and *MuDRA* algorithms are built in the Django framework. The back-end utilizes a Postgres [6] database and interfaces with Django [7]. Users can tune the algorithmic parameters at any given time on the front-end. The front-end periodically relays the parameters to Django. Django utilizes the user input and system state information derived from the back-end to run the required FB and rate adaptation algorithms. The system state is then relayed to Angular, which renders the information on user's screen. Finally, the experiment can be paused at any time to allow the video tool to generate videos at the nodes for that period of time. The video tool uses ffmpeg to render and generate the videos and an nginx server [8] to transmit to the front-end.

Fig. 2 shows a screenshot of the application. Demo participants can select different experiment settings on the web interface. This information is used along with data collected from the experiments to show how the performance at all the nodes on the grid. The feedback nodes are highlighted with a red border. The rate adaptation and multicast throughput measured at the AP appears below.

V. Acknowledgements

This work was supported in part by NSF grant CNS-10-54856, CIAN NSF ERC under grant EEC-0812072, and the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme (FP7/20072013) under REA grant agreement no. [PIIF-GA-2013-629740].11.

References

[1] "Cisco, white-paper, cisco connected stadium Wi-Fi solution," 2011. [Online]. Available: http://www.cisco.com/web/strategy/docs/sports/c78-675064_svcs.pdf
[2] R. Chandra, S. Karanth, T. Moscibroda, V. Navda, J. Padhye, R. Ramjee, and L. Ravindranath, "DirCast: a practical and efficient Wi-Fi multicast system," in *Proc. IEEE ICNP'09*, 2009.
[3] Y. Bejerano, J. Ferragut, K. Guo, V. Gupta, C. Gutterman, T. Nandagopal, and G. Zussman, "Scalable WiFi multicast services for very large groups," in *Proc. IEEE ICNP'13*, 2013.
[4] V. Gupta, C. Gutterman, Y. Bejerano, and G. Zussman, "Experimental evaluation of large scale WiFi multicast rate control," in *Proc. IEEE INFOCOM'16*, 2016.
[5] "AngularJS." [Online]. Available: https://angularjs.org/
[6] "PostgreSQL." [Online]. Available: http://www.postgresql.org/
[7] "Django project." [Online]. Available: https://www.djangoproject.com/
[8] "NGINX." [Online]. Available: https://www.nginx.com/resources/wiki/