A Demonstration of Multirate Multicast Over an 802.11 Mesh Network

Georgios S. Paschos*‡, Chih-Ping Li*, Eytan Modiano*, Kostas Choumas§ and Thanasis Korakis§

*LIDS, Massachusetts Institute of Technology, Cambridge, MA, USA
§Dept. of ECE, University of Thessaly, Volos, Greece
‡Informatics & Telematics Institute, CERTH, Greece

***Abstract*—This demo presents a novel multirate multicast scheme for video delivery to wireless users. We demonstrate an adaptive scheme that combines differential backlog scheduling and intelligent packet dropping, both based on local information. An important feature of this scheme is that it does not require centralized calculations. We focus on 802.11 mesh networks and we show that our solution copes efficiently with the volatile nature of the wireless medium. We demonstrate how a video multicast from a single point to multiple receivers could utilize layered multimedia coding techniques to adapt efficiently users' perceived quality to their allowable data rates.**

I. Introduction

We demonstrate Maximum Multicast Utility for Wireless (MMU-W) policy, a multicast algorithm for delivering multimedia streams to multiple users across a wireless mesh network. Since different receivers may require different data rates, we study in [1] the problem of per receiver network utility maximization in multirate multicast, where each receiver is assigned a potentially different utility function. The presented solution combines backpressure-type scheduling with intelligent threshold-based packet dropping at intermediate nodes to optimally solve this problem. Packets corresponding to all stream layers are initially injected into the network without any calculations, which makes the approach very lightweight and hence desirable for wireless systems. Progressively, some packets are dropped according to a dropping scheme which bases its decisions on local information. We combine the above with receiver-end flow control to produce a scheme that maximizes utility without source cooperation. We show that the original stream is stripped of unnecessary packets so that each receiver obtains the exact amount of information that corresponds to maximum aggregate utility. The routing of each session is based on a fixed multicast tree. Next, we propose the main outlines of the MMU-W policy.

Each node maintains one transmission and one drop queue for each outgoing link and multicast session. Since the injected video rate might be higher than the wireless link capacity, our algorithm may need to discard some packets. Following a decision to discard a packet, the packet is first moved to the drop queue before it is ultimately discarded. The time is virtually slotted at every node, although the nodes are not assumed to be synchronized. During each slot the following actions are performed at each node:

- *Scheduling:* Each node chooses one of its outgoing links to activate and allocates the whole rate of this link to a session, in a way that this link-session choice features the maximum product of link rate and weighted differential backlog.
- *Packet Dropping:* A predefined number of packets is moved from each transmission queue to its corresponding drop queue, if the former is longer than the latter one. Then, the drop queue removes the same amount of packets from the network, if it exceeds a predefined threshold. This process protects the backlogs from overflowing while at the same time it is used to optimize long-term flow rates.
- *Receiver-End Flow Control:* Each destination features a pressure that is the length of a virtual queue, which indicates the urgency of the particular receiver to obtain more or less packets according to its utility. This pressure affects the scheduling decisions, so that a destination with high positive pressure is very unlikely to receive new packets.

II. Demonstration Setup

To demonstrate the practicality of the MMU-W policy, we develop a prototype implementation in NITOS testbed [2]. NITOS is a heterogeneous outdoor testbed, where two types of networks are used: a wireless network with IEEE 802.11a/b/g/n protocol and a wired network using Gbit Ethernet. Being partly deployed in a building roof, NITOS is a non-RF-isolated wireless testbed. To eliminate interference we employed 802.11a, which is not used by commercial 802.11 products in Greece. The NITOS nodes feature a 3.4GHz Intel i7 processor and two Atheros wireless cards.

The implementation is based on the Click Modular router framework [3]. Click facilitates experimentation and evaluation of scheduling and flow control algorithms in real systems. It runs as a user-level daemon at each node and via the libpcap library it provides full control on packet transmission. Our implemented framework includes mechanisms for estimating channel quality, forming a queue structure, exchanging queue backlog information, and splitting time into virtual slots.

- *Estimating Channel Quality.* To evaluate channel quality, we adopted the ETT estimation algorithm of Roofnet [4]. Nodes periodically broadcast probes which are used to estimate the successful transmission probability. With this process every node periodically obtains a table with the qualities for each channel rate/neighbor pair. This mechanism is known to incur negligible throughput overhead [4].
- *Queue Structure.* We implement the transmission queues on each node and we create counters for the drop and virtual queues that indicate their lengths. Each node utilizes multiple transmission queues for packet storage, one per each couple of session and outgoing link. The counter of the virtual queue may take non-integer values. Each of these internal queues/counters is created upon the arrival of the first packet of a new session.

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

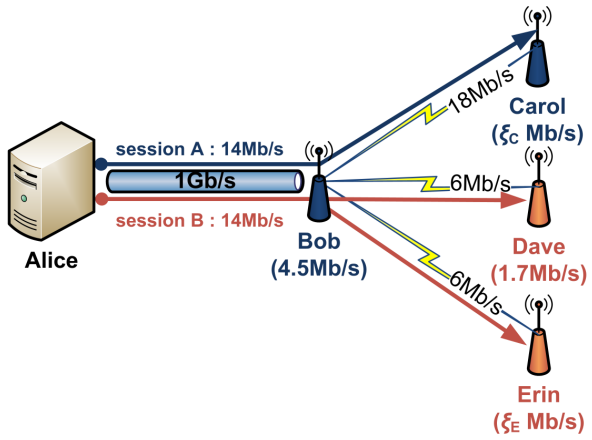

Fig. 1: Experiment topology with five NITOS nodes. Two sessions A and B are generated at Alice, forwarded to Bob via a wired connection, and then distributed to Carol, Dave, and Erin through wireless. The Figure shows the rate requirement per receiver (in parentheses) and the physical rate per link.

• *Exchanging Queue Backlog Information.* To compute the differential backlogs, each node broadcasts periodically the backlog size of all its transmission queues. If a node is a destination for some session, it also broadcasts the backlog size of its virtual queue. The broadcast messaging is repeated once every second. Prior experiments suggest that more frequent broadcasts incur visible throughput overhead, while rarer broadcasts may affect the delay performance due to obsolete queue information.

• *Virtual Slots.* Each node keeps an internal timer that expires once every slot. Upon counter expiration the policy selects the next queue to be served and for the duration of the next slot the decision remains fixed. The slot duration is set to 100msecs, equal to 1/10 of the broadcasts period. Small values for the slot duration improve delay and reduce throughput fluctuations but burden the CPU of the device.

III. Demonstrated Scenarios

We conduct experiments on the specific topology of Figure 1. Five NITOS nodes are used: Alice and Bob are connected via Ethernet while Bob is connected to the other three nodes via wireless. The nodes are configured to run the MMU-W policy. The wireless links use fixed physical rates instead of the 802.11 rate adaptation scheme. In particular we set the physical rates to 18Mb/s, 6Mb/s and 6Mb/s for the links to Carol, Dave, and Erin respectively. The physical rate of the wired connection is 1Gb/s.

We consider two sessions, A and B, each with traffic rate 14Mb/s. The source node for both sessions is Alice and the multicast receivers are {Bob, Carol} for A, and {Dave, Erin} for B. To generate packets we use two UDP streams created with the Iperf tool [5]. We run the Iperf tool on external nodes to avoid polluting the CPU measurements. The receiver rate requirements are 4.5Mb/s for Bob, ξ_CMb/s for Carol, 1.7Mb/s for Dave and ξ_EMb/s for Erin, where the values ξ_C, ξ_E are chosen differently per experiment. The objective is to satisfy all receiver rate requirements as well as achieve maximum throughput.

In Figures 2 and 3, we show the measured instantaneous and average throughput for two scenarios. The instantaneous throughput is computed as the average over 1sec periods. In the first scenario we choose $(\xi_C, \xi_E) = (2.8, 1.7)$, see Figure 2. The objective is achieved because all receiver requirements are satisfied and the excess wireless resource is allocated to the receiver with the highest capacity, i.e. Carol. We observed that the wireless medium was fully utilized. In the second scenario, we reverse the requirements of Carol and Erin, $(\xi_C, \xi_E) = (1.7, 2.8)$, see Figure 3. The theoretical total throughput is smaller in this case due to Erin's low physical rate and high requirement. We observe the user-level CPU occupancy using the vmstat command. The occupancy is 8-10% for the whole duration of experiments, which is encouraging. The CPU usage was the same at all nodes, indicating that our policy does not incur extra burden on the sources. Additionally, it was largely independent of data rates used, which implies that packet operations and queue maintenance have a minor contribution to the CPU occupancy.

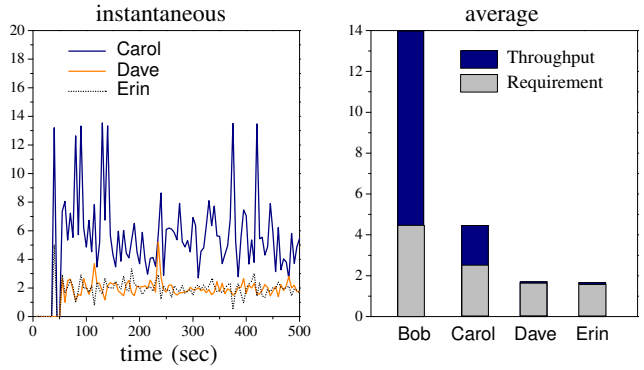

Fig. 2: Scenario 1: $(\xi_C, \xi_E) = (2.8, 1.7)$. Instantaneous and average throughput (Mb/s) are shown.

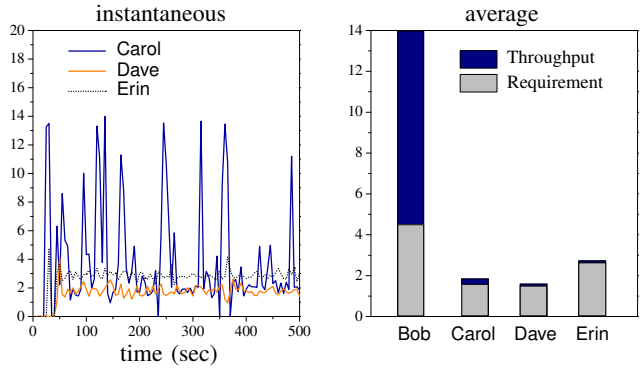

Fig. 3: Scenario 2: $(\xi_C, \xi_E) = (1.7, 2.8)$. Instantaneous and average throughput (Mb/s) are shown.

References

[1] G. S. Paschos, C.-P. Li, E. Modiano, K. Choumas, and T. Korakis, "Multirate Multicast: Optimal Algorithms and Implementation," *IEEE Infocom*, 2014.

[2] "NITLab: Network Implementation Testbed Laboratory," http://nitlab.inf.uth.gr/NITlab/index.php/testbed.

[3] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM SOSP*, 1999.

[4] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network," *MobiCom*, 2005.

[5] "Iperf: The TCP/UDP Bandwidth Measurement Tool," http://sourceforge.net/projects/iperf/.