Towards Optimal MAC without Message Passing in Wireless Networks *

Yung Yi, Jinsung Lee, Song Chong
EECS, KAIST
{yiyung,jinsl,song}@ee.kaist.ac.kr

Mung Chiang
Princeton University
chiangm@princeton.edu

Alexandre Proutière
Microsoft Research Lab, UK
alexandre.proutiere@microsoft.com

ABSTRACT

One of the key messages in significant research on cross-layer design of wireless networks over the past 15 years is that extensive message passing may be necessary to achieve optimal performance. Heavy message passing is obviously an undesirable feature for practical implementation due to its reduced actual throughput and security issues. In this paper, we first summarize our preliminary work that does not need message passing, yet achieves near-optimality. Then, we present our recent efforts to realize such an algorithmic solution in a real network, e.g., IEEE 802.11-based systems, and discuss possible future research directions.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

Wireless Networks, Max-Weight Scheduling, CSMA, Message Passing, 802.11 Implementation

1. INTRODUCTION

There have been growing interests in the design of scheduling algorithms which efficiently and fairly exploit the radio resources in wireless networks in recent years. In their seminal work [1], Tassiulas and Ephremides developed a centralized scheduling algorithm, the Max-Weight scheduler, achieving throughput optimality. The traffic scenario considered in [1] is that of infinite buffers fed by exogenous random packet arrivals with fixed rates, and being throughput optimal means that the proposed algorithm achieves the maximum stability region. In this paper, we are interested in a different traffic scenario, more appropriate to represent the elasticity of traffic in data networks. We considered saturated users (i.e., with fully back-logged buffers), who perceive performance as a function of the average service rate, and the problem is to design a scheduling algorithm that achieves the desired trade-off between efficiency and fairness. Specifically, the proposed scheduling algorithm aims at maximizing the sum of user utilities, where the utility $U(\cdot)$ is a non-decreasing and concave function of the user average service rate, formulated by:

$$\begin{array}{ll} \text{maximize} & \sum_{l\in\mathcal{L}} U(\gamma_l) \\ \text{subject to} & (\gamma_l : l \in \mathcal{L}) \in \Lambda, \end{array} \quad (1)$$

where $\mathcal{L}$ is the set of links, Λ is the link-level maximum stability region, and γ_l is the long-term throughput achieved over link l.

This optimization problem has received a lot of attention recently, for it appears not only as a MAC-layer problem but also it can be extended to joint rate control and scheduling through dual decomposition (see [2] and the references therein). The key message in cross-layer design involving congestion control, routing, and MAC scheduling is that MAC scheduling may be the main bottleneck to practical implementation. Motivated by it, there has been a long series of work on distributed scheduling, involving randomization, maximal/greedy scheduling, and random access with message passing. They usually require some information of the queues to be passed around among the nodes (see [3] and the references therein). These signaling overhead reduces the "effective throughput" and makes the algorithms not fully distributed. This naturally leads to the following question that turns out to be very challenging: *what about the performance of random access algorithms without any message passing?*

In recent papers, it has been demonstrated that random access could also achieve high throughput performance. For example, in [4–6], it has been shown that non-adaptive CSMA (Carrier Sense Multiple Access) algorithms, where each link accesses the channel with a *fixed* probability, are able to provide average throughput close to throughput-optimality. Turning to random access with adaptive channel access rate, [7] first proposed a simulated-annealing based approach to distributed scheduling. A similar idea has been developed this and last year in three papers at similar time [8–10].

In this paper, we first summarize our work [10] over multi-hop wireless networks with only single-hop sessions, which achieves utility-optimality without message passing developed from the theoretical point of view. We then discuss our research effort to realize such an algorithmic solution

*This work was partly supported by IT R&D program of MKE/IITA [2009-F-045-01] and Korea Research Council of Fundamental Science and Technology.

CFI '09 June 17–19, 2009, Seoul, Korea.

in the real 802.11 based wireless network, as well as further extensions to multi-hop sessions and delay issues.

2. ADAPTIVE CSMA

Consider a wireless network composed by a set $\mathcal{L}$ of L interfering links. Interference is modeled by a symmetric, boolean matrix $A \in \{0,1\}^{L\times L}$, where $A_{lk} = 1$ link l interferes with link k. The transmitter of link l can transmit at a fixed unit rate when active, and all links are saturated with infinite backlog. We consider only link-level single-hop sessions.

To access the channel, each transmitter $l \in \mathcal{L}$ runs a random back-off algorithm parameterized by two positive real numbers (λ_l, μ_l), denoted as CSMA(λ_l, μ_l): after a successful transmission, the transmitter randomly picks a back-off counter according to some distribution of mean λ_l; it decrements the counter only when the channel is sensed idle; and it starts transmitting when the back-off reaches 0, and remains active for a duration μ_l, i.e., channel holding time. If parameters $(\lambda_l, \mu_l), l \in \mathcal{L}$ were fixed, the analysis of the dynamics of *continuous-time* CSMA algorithms would be classical (e.g., [5] and references therein). In this case, in steady state the set of active links evolves according to a *reversible Markov process*.

We now describe how transmitters adapt their CSMA parameters. Time is slotted and transmitters update their parameters at the beginning of each slot. To do so, they maintain a virtual queue, denoted by $q_l[t]$ in slot t, for link l.

A-CSMA Algorithm

1. During slot t, the transmitter of link l runs CSMA$(\lambda_l[t], \mu)$, and records the amount $S_l[t]$ of service received during this slot;
2. At the end of slot t, it updates its *virtual* queue and its CSMA parameters according to:

$$q_l[t+1] = \left[q_l[t] + b[t]\left(\frac{V}{q_l[t]} - S_l[t]\right)\right]_{q^{\min}}^{q^{\max}}, \qquad (2)$$

and set

$$\lambda_l[t+1] = \mu^{-1}\exp(q_l[t+1]). \qquad (3)$$

At the beginning of each slot, each non-active transmitter picks a new random back-off counter λ_l to account for the CSMA parameter update. In **A-CSMA**, $b : \mathbb{N} \to \mathbb{R}$ is a step size function; V, $q^{\min}$, $q^{\max}(> q^{\min})$ are positive parameters, and $[\cdot]_c^d \equiv \min(d, \max(c, \cdot))$. Note that the fixed μ does not change the main results dut to its insensitivity to the distribution of channel holding time.

We can prove that the long-term averaged throughput by **A-CSMA** algorithm asymptotically converges to the optimal one, i.e., it is the solution of (1). By asymptotically, we mean that the throughput by **A-CSMA** can be made arbitrarily close to the optimal one, by increasing V arbitrarily large. We refer the readers to the theorems and the proofs in [10] for more details.

The main difficulty in analyzing the convergence of **A-CSMA** lies in the fact that the updates in the virtual queues depend on random processes $(S_l[t], t \geq 0)$, whose transition rates in turn depend on the virtual queues. The major intuition behind optimality is as follows: for sufficiently small step size $b[t]$, the time-scale of $q[t]$ becomes much slower than that of $S_l[t]$, so that the random access part observes $q[t]$ that is almost constant. Then, from the theory of reversible Markov process as well as the update rule in (3), the stationary distribution of random schedule $(S_l[t] : l \in \mathcal{L})$ is such that the probability of Max-Weight scheduling is arbitrarily close to 1, by enlarging V. Thus, it turns out that **A-CSMA** selects a schedule with max-weight infrequently. Note, however, that the proof of convergence to the optimal solution of (1) is far from trivial, since the queue lengths still change before **A-CSMA** reaches a stationary regime. The update rule in (3) states that links need to aggressively access the channel when their queues are large.

3. CURRENT WORK AND FUTURE DIRECTIONS

3.1 Implementation in 802.11 Network

We now describe how our theory can be transferred to practice, where, different from the ideal continuous model in the previous section, transmitters run discrete back-off algorithms, so that time is slotted and the back-off window must be greater than one slot. Roughly speaking, we use the *access probability*, denoted by $0 \leq p_l[t] \leq 1$, to emulate $\lambda_l[t]$ in the continuous model. However, by nature of the back-off window having one slot at minimum, $\lambda_l[t]$ (resp. $p_l[t]$) cannot be set to be arbitrarily large (resp. small) due to large $q_l[t]$ values. To tackle such issues, we use large values of channel holding time μ, and choose $p_l[t]$, such that $p_l[t] \times \mu = \exp(q_l[t])$. The choice of μ depends on the chosen value of q_{max}.

As a practical system, we consider a popular IEEE 802.11-based system. Our choice of 802.11 as a target system is more motivated by the fact that their device drivers are open so that we can fully control their operation parameters to implement our theory-driven algorithm. There are two key issues to be addressed: *(i)* adjusting backoff counter and *(ii)* controlling channel holding time.

Adjusting backoff counter. In 802.11 systems, whenever a node has a packet to transmit, it generates a random *back-off counter* uniformly from $[0, CW-1]$, where CW corresponds to the contention window size. If channel is idle for a DIFS (Distributed Inter Frame Space), the node begins to decrement its back-off counter by one for each idle slot. If channel is busy, the back-off counter is frozen until the next idle DIFS is sensed. When the back-off counter reaches 0, the node begins a transmission. A node assures its successful transmission from receiving an ACK. Packet transmission failure (mainly due to collision) forces nodes to double CW, and the retransmission with the doubled back-off will be attempted up to the pre-defined retry limit. See [11] for more details on 802.11.

To implement our algorithm, we make the following changes:

1) We disable the binary exponential back-off and also set the retry limit to be zero to nullify the default back-off mechanism of 802.11;
2) in 802.11, CW is maintained at each node, not each link, i.e., one contention window per interface. We modify the device driver at MAC, such that a separate contention window can be associated with each outgoing link, which we denote by CW_l;
3) the link l emulates the channel access probability p_l by suitably choosing the maximum contention window size

CW_l (and then randomly picking a slot to access):

$$CW_l = \frac{2}{p_l}, \tag{4}$$

where '2' is needed since the actual contention window size is selected randomly from $[0, CW_l - 1]$.

Controlling channel holding time μ**.** In 802.11, two kinds of sensing schemes are used to determine the state of medium: physical and virtual CS (Carrier Sensing) [11]. When either indicates a busy medium, the medium is considered busy. The physical and virtual CSs are provided by different layers, PHY and MAC. To reserve the medium, NAV (Network Allocation Vector) is used: The NAV maintains prediction time for future arriving traffic according to the duration information in the MAC headers. Nodes receiving a valid packet update their NAV with the new value that is greater than the current value. Note that the NAV is *not* updated for the packet destined to the node itself. We exploit this feature to control channel holding time. When a link wants to extend its holding time, its transmitter sends a packet destined to its receiver with a large NAV value enough to transmit the subsequent packets during a given holding time. The other nodes receiving this packet remain silent until their NAV reaches 0. In this manner, only one link can be activated during the holding time μ without interference from other links in its interference range.

3.2 Simulation Results

Prior to the field tests, in order to validate our algorithm and further identify the possible issues, we have conducted simulations using GloMoSim simulator citation.... We implemented our testbed that allows GloMoSim code can be applied to 802.11 systems without change, referred to as Common Code Architecture (CCA), see xxx for details. We plan to make real experiments using CCA in the near future.

We consider a grid network with 6 nodes, as shown in Fig. 1, where the dotted lines represent the connectivity. Here, flow 2 interferes with both flows 1 and 3. We use the Proportional-Fairness as source rate controllers, i.e., $U(\cdot) = \log(\cdot)$. In our setup, at the Proportional-Fairness, $\gamma_2^* = 1/3 \approx 0.33, \gamma_1^* = \gamma_3^* = 2/3 \approx 0.66$ are optimal.

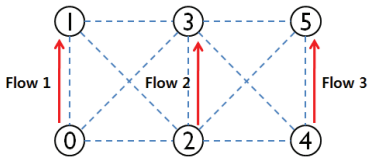

Figure 1: A simple network with three flows

Fig. 2 shows the relationship between efficiency and holding time. First, we observe efficiency gap from the ideal continuous case, because (i) idle slots of waiting for channel access due to finite intensities, and (ii) packet collisions due to discretization. Second, we see that increasing holding time leads to increasing efficiency. However, the efficiency increase becomes less sharp as the holding time becomes larger, see xxxx. We also make an interesting observation that in the discretized system, the efficiency loss of the links with larger collision domain experience more severe degradation than those with smaller collision domain (e.g., link 2 vs. links 1 and 3). We conjecture that link 2 turns out to be more affected by emerging collisions generated by transition from continuous without collisions to discrete with collisions. Currently, we are rigorously analyzing this case.

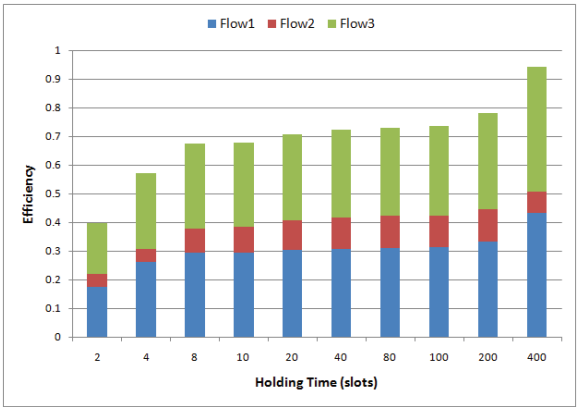

Figure 2: Aggregate efficiency of three flows

Fig. 3 and 4 represent efficiency and total backlog (implicitly telling us delay from Little's law) with changing V of (2). As V increases, the efficiency is gradually improved while still flow 2 experience more efficiency loss. Observe that the parameter V is responsible for controlling trade-off between efficiency (i.e., throughput) and backlog (i.e., delay), e.g., see [14].

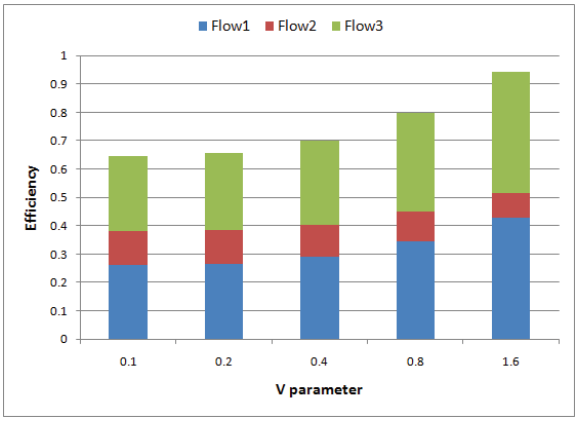

Figure 3: Aggregate efficiency of three flows

3.3 Extensions and Testbed

We plan to extend our theory-driven **A-CSMA** to a more complete protocol in real systems by extending it to a version that can deal with multi-hop sessions as well as by adding more ideas to reduce delays.

Multi-hop sessions. Currently, we have considered only single-hop sessions. More practical scenarios will include the case when a session is configured with the notion of end-to-end flows that traverse multi-hop wireless links. In that case, routing should be jointly considered with our **A-CSMA** algorithm. We believe that this extension is somewhat straightforward in theory, by either (i) assuming the routes are given by routing layer, or (ii) leaving routing functions as another degree of freedoms. We currently consider a back-pressure based routing and congestion control [1].

Delay Reduction. With end-to-end multi-hop flows, they go through multi-hop paths and each link along those paths

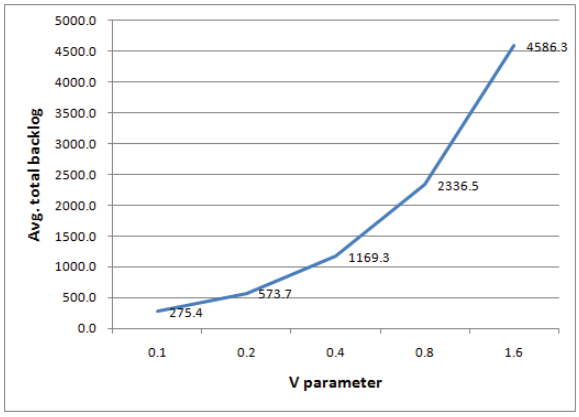

Figure 4: Average total backlog of three flows

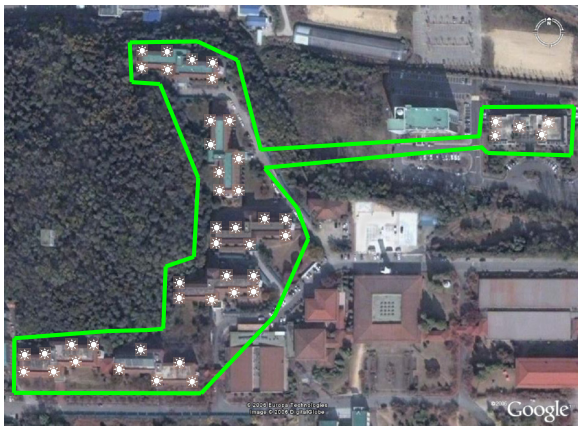

Figure 5: WiSEMesh: Campus-wide Wireless Mesh Network Testbed at KAIST [15]

needs to keep many packets in the queue, their delay performance may be very poor depending on routing/scheduling strategy. Especially in back-pressure based scheme, the network signifies source nodes of congestion by indirectly "back-pressing" congestion information (from the congestion point to the corresponding source node). This may take long time and require a large number of packets to be buffered at the queues of intermediate nodes. In recent works [12, 13], the authors proposed the enhanced algorithms to decrease delay by considering shortest path based routing or just emulating the back-pressure-driven queues. However, they focused on the Max-Weight rule as a MAC scheduling. Thus, as a next step, it remains to understand what happens to delay performance if much simpler MAC such as our **A-CSMA** is used. We believe that all these efforts to consider multi-hop sessions as well as to reduce delay are a crucial key step to practical cross-layer designs that are *simple* as well as *optimal* in performance.

Implementation at KAIST Testbed. We are currently implementing and extending the **A-CSMA** in our campus-wide testbed, WiSEMesh [15]. The WiSEMesh has 55 nodes deployed in the campus area providing Internet connection for hundreds of users. Each node is a small linux-based host with more than two wireless NICs based on 802.11 technology. We developed the WiSEMesh node software stack that contains linux OS, wireless NIC drivers, tools such as DHCP server, NAT and etc. Fig. 5 shows its deployment map. Through modification mentioned in section 3.1 as well as multi-hop sessions and delay reduction section, our proposal can be implemented easily on top of the traditional 802.11 protocol. The testbed implementation in this real systems is expected to verify our theories as well as give positive feedback toward practical, simple, yet, near-optimal MAC protocol that takes a large step to nice and practical cross-layer design in wireless networks.

4. REFERENCES

[1] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1949, Dec., 1992.

[2] Y. Yi and M. Chiang. Stochastic network utility maximization. *European Transactions on Telecommunications*, Mar., 2008.

[3] Y. Yi and A. Proutiere and M. Chiang. Complexity in Wireless Scheduling: Impact and Tradeoffs. In *Proc. of ACM Mobihoc*, 2008.

[4] A. Proutiere and Y. Yi and M. Chiang. Throughput of Random Access without Message Passing. In *Proc. of CISS*, 2008.

[5] M. Durvy and P. Thiran. Packing Approach to Compare Slotted and Non-slotted Medium Access Control. In *Proc. of IEEE Infocom*, 2006.

[6] C. Bordenave and D. McDonald and A. Proutiere. Performance of Random Multi-access Algorithms, an asymptotic approach. In *Proc. of ACM Sigmetrics*, 2008.

[7] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, 1988.

[8] J. Shin and D. Shah and S. Rajagopalan. Network adiabatic theorem: An efficient randomized protocol for contention resolution. In *Proc. of ACM Sigmetrics*, 2009.

[9] L. Jiang and J. Walrand. A CSMA Distributed Algorithm for Throughput and Utility Maximization in Wireless Networks. *Tech. Report, UCB*, 2008.

[10] J. Liu and Y. Yi and A. Proutiere and M. Chiang and H. V. Poor. Maximizing Utility via Random Access Without Message Passing. *Tech. Report, Microsoft Research Labs, UK*, 2009.

[11] IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications. http://www.ieee802.org/11/.

[12] L. Bui and R. Srikant and A. Stolyar. Novel Architectures and Algorithms for Delay Reduction in Back-pressure Scheduling and Routing. In *Proc. of IEEE Infocom*, 2009.

[13] L. Ying and S. Shakkottai and A. Reddy. On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks. In *Proc. of IEEE Infocom*, 2009.

[14] M. Neely. Super-Fast Delay Tradeoffs for Utility Optimal Fair Scheduling in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1489–1501, Aug., 2006.

[15] WiSEMesh at KAIST. http://netsys.kaist.ac.kr/wisemesh/doku.php.