Meta-MAC Protocols: Automatic Combination of MAC Protocols to Optimize Performance for Unknown Conditions

András Faragó, *Member, IEEE*, Andrew D. Myers, *Student Member, IEEE*, Violet R. Syrotiuk, *Member, IEEE*, and Gergely V. Záruba, *Student Member, IEEE*

***Abstract*—A *systematic* and *automatic* method to dynamically combine any set of existing MAC protocols into a single higher layer, or *meta*-MAC protocol, is presented. The new approach makes it possible to always achieve the performance of the best component protocol, without knowing in advance which protocol will match the potentially changing and unpredictable network conditions. Moreover, this dynamic optimization is entirely automatic and runs without any centralized control or any exchange of messages, using only local network feedback information. We describe the method and prove that the resulting meta-MAC protocol achieves optimal performance in a well-defined sense. Through simulation on different types of networks and with different component MAC protocols, we demonstrate that our simple and practical combination algorithm yields highly adaptive and scalable MAC solutions.**

***Index Terms*—Access protocols, adaptive systems, distributed algorithms, multiaccess communication, optimization methods.**

I. Introduction

IN ALL NETWORKS that have a broadcast channel as the basis of communication, the medium access control (MAC) protocol serves a vital role. It is the MAC protocol that is directly responsible for controlling access to the communication resources. There are seemingly countless MAC protocols, each optimized for specific network conditions. The network designer naturally faces the question: which one to use? Even if the network conditions are known precisely in advance, the answer is very often not easy, due to the large number of competing protocols. In most cases, however, the designer does not even know the exact network conditions and/or has to assume that they may change during operation, usually with limited predictability. For example, in mobile multihop networks, the topology changes frequently, due to node movement. But even in a fixed and fully connected network, the traffic pattern can be very unpredictable and unstable.

The usual approach to handle unknown or changing conditions in most MAC protocols is to include some kind of *adaptivity* in order to adjust the operation to the actual network conditions. There are numerous ways known to make MAC protocols adaptive, including various handshake mechanisms to avoid collisions, monitoring traffic intensity in order to change strategies, as well as many other ad hoc solutions.

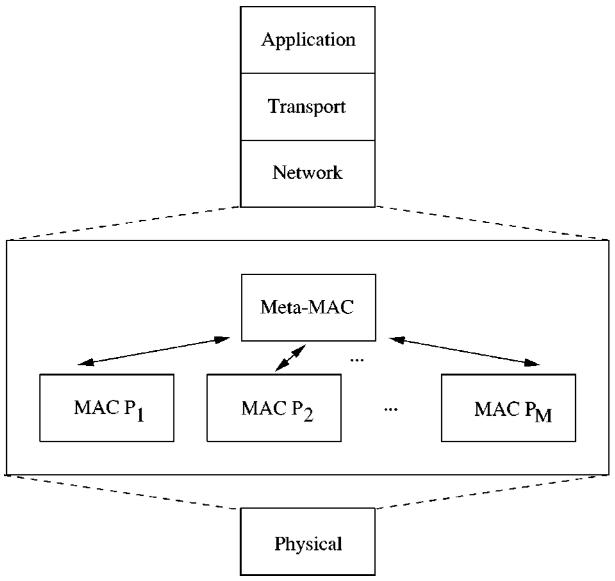


Fig. 1. The meta-MAC protocol in a simplified protocol stack.

We propose a principally new approach to create adaptive and scalable MAC solutions. Our new approach is based on a "meta-MAC" protocol framework that implements a *higher layer* of adaptivity, on top of the existing MAC protocols. Specifically, we introduce a method to *systematically* and *automatically* combine any set of existing protocols into a single MAC protocol such that the resulting combined protocol has provable optimality properties. Thus, we assume that a number of existing MAC protocols are available as *components* at each node in the network, and that our meta-MAC protocol works on top of them (see Fig. 1), optimally combining their individual transmission decisions into a final decision at the node each time when such a decision has to be made. Then, according to the local network feedback information, the combination is updated. The resulting combination may not be identical at each node of the network; this depends on whether or not each node receives the same feedback.

It is important that the meta-protocol can combine any set of component protocols *automatically*. Thus, we can select appropriate component protocols that are already fine-tuned for certain network conditions and may also have internal adaptivity

Manuscript received October 7, 1999; revised March 1, 2000.

The authors are with the Department of Computer Science and Center for Advanced Telecommunications Systems and Services (CATSS), University of Texas at Dallas, Richardson, TX 75083-0688 USA (e-mail: farago@utdallas.edu; amyers@utdallas.edu; syrotiuk@utdallas.edu; zaruba@utdallas.edu).

Publisher Item Identifier S 0733-8716(00)07123-7.

properties. Each component may be a good candidate for certain situations. For example, a contention protocol is good for low loads, due to its low delay, while a TDMA protocol is desirable for high loads, as it avoids the breakdown induced by too many collisions. Then the meta-protocol will automatically find combined decisions that dynamically represent the "best of the team," under the actual network conditions, without having to know in advance which of the conditions will actually occur and how they will change. Moreover, the optimization runs locally without any centralized control or any message exchanges. Thus, what we present is *not* just another MAC protocol. On the contrary, it is a methodology that allows the *optimal aggregation* of several existing protocols in a unique, scalable way, so that they complement each other and result in an overall increase in network efficiency.

Our proposed combination principle is practical to implement even in a mobile multihop wireless environment, which is normally considered a difficult scenario, due to the lack of full connectivity. For example, the radios in Raytheon Systems ASPEN project [16] have the capability to simultaneously load multiple MAC protocols and the ability to switch protocols and tune parameters on the fly. Our meta-protocol has very low complexity, lending itself to implementation in hardware. In light of the performance gains that the meta-protocol achieves, we believe the changes to the network equipment are both justified and practical to implement.

The rest of this paper is organized as follows. In Section II we overview some precursors of protocol combination, including dynamic parameter optimization as a simple form. Section III describes our systematic meta-MAC protocol framework and what claims can be made regarding its optimality. For clear presentation of the fundamental principle, in this paper we restrict ourselves to slotted time and assume that perfect feedback is available at the end of each slot. In Section IV some examples of the principle for LANs and multihop networks are described, supported by simulation results. We also illustrate how the principle can be used to optimize protocol parameters. Section V concludes the paper.

II. Precursors of Protocol Combination

There are many examples of MAC protocols being combined together to enhance adaptivity and performance. The combination in these protocols, however, is done in an *ad hoc* way without systematic optimization. Sometimes, the combination is "hidden," i.e., without explicitly referring to component protocols. One such family of examples is the p-persistent slotted Aloha protocols where in each slot, whenever there is a packet in the queue, the probability of transmission is a constant p independently for each slot. If we dynamically change the value of p in any way, e.g., with binary exponential backoff (BEB), then essentially we combine different p-persistent slotted Aloha protocols that differ in their p values. Thus, in each slot we decide to use one of these component protocols, namely the one with the appropriate p.

In the above example, it is not an easy question which is the best way of adjusting the retransmission probabilities. It is proven that BEB results in an unstable protocol under certain modeling assumptions [1]. The existence of stable protocols in this setting also depends on the type of feedback available from the channel and on how the user population is modeled. For acknowledgment-based protocols, a large class of backoff schemes, including polynomial backoff[1] is unstable [10]. In contrast to this, for finite user population, any superlinear polynomial backoff protocol has been proven stable, while BEB still remains unstable above a certain arrival rate [7]. Thus, it is far from trivial to find the best combination.

In [12], a learning automata based random access protocol for WDM passive star networks is introduced to optimize the transmission probability of each wavelength. This work is related to our meta-protocol approach that we present in the next section; however, it solves only one specific parameter optimization problem. Our approach is far more general and applicable to any network that has a broadcast channel as the basis of communication.

In mobile multihop networks, *spatial reuse* TDMA protocols can adapt by periodically reassigning time slots and frame lengths (see, among many others, [6] and [17]). In these protocols, the nodes alternate between a contention and a TDMA protocol. The contention protocol is used by the nodes to create TDMA schedules. Once the schedules are fixed, the operation switches over to the TDMA protocol. When node mobility results in a topology change, the contention protocol runs again.

Another way of combining protocols is based on nodes monitoring the traffic intensity of the medium locally, e.g., by counting the number of idle slots in a frame. This measurement can be used to determine the rate at which a node can transmit packets [9], trigger the node to switch protocols in the next frame [2], or otherwise decide to have nodes contend in some slots [11].

A more explicit example of protocol combination is the idea of *protocol threading*, first used with TSMA protocols in mobile multihop networks [4]. In this approach, several different TSMA protocol frames are interleaved on a time sharing basis to obtain a *threaded TSMA* protocol. In this solution, the transmission rights are assigned in different time slots according to different TSMA protocols in a cyclically repeated way, realizing a time sharing that yields a combined protocol with unique properties [4]. The advantage of this combination is that the component TSMA protocols are optimized for different densities of the topology, and the threaded protocol can handle all situations without knowing in advance which one will occur.

Another approach to combining protocols is the ADAPT protocol of [5] where, in principle, any allocation protocol can be combined with any contention protocol, such that the allocation protocol provides guaranteed access, while the contention protocol utilizes the unused slots to enhance performance.

III. The Meta-MAC Protocol

In this section we propose a systematic and automatic principle to combine MAC protocols via a specific meta-MAC protocol. The meta-MAC protocol is completely general in the sense that it can be applied in any network that has a broadcast

[1]The backoff interval grows according to a polynomial function rather than an exponential function.

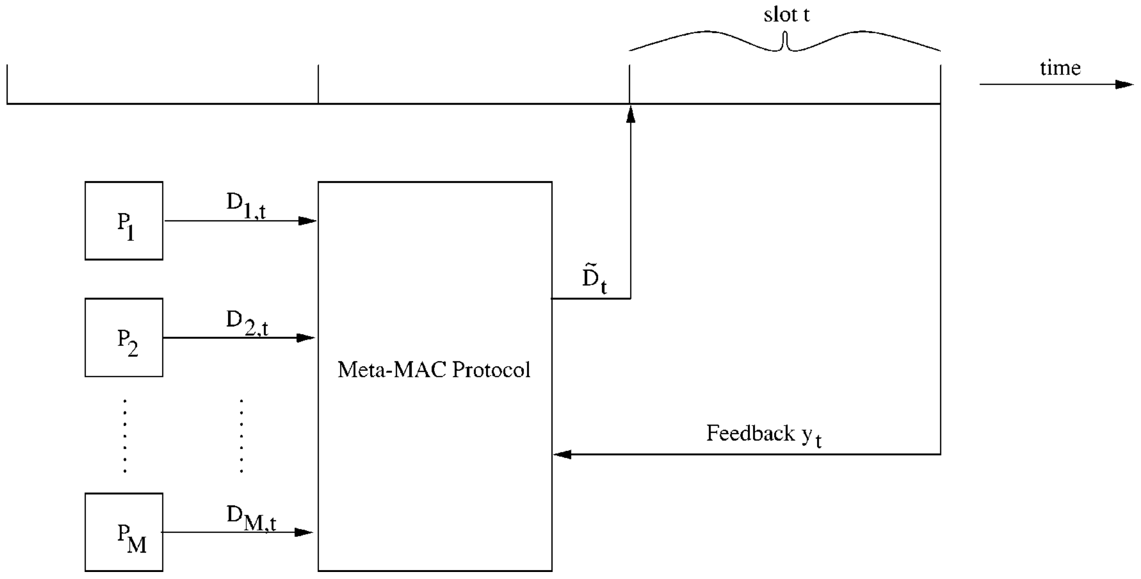


Fig. 2. Operation of the meta-MAC protocol.

channel as the basis of communication, and it can run on top of any set of component protocols. For ease of presentation, we restrict our attention to slotted time and assume that perfect feedback is available at the end of a slot. The actual way of computing the combined transmission decision in each time slot is based on a weighted combination of the individual decisions of the component protocols, with rounding the weighted average at the end to obtain a binary decision, using randomization. The weights are then appropriately adjusted after each slot, based on the continuously updated "credit history" of the individually running component protocols using local network feedback information. We call the method the *randomized weighted majority (RWM) meta-MAC* protocol. After introducing the method, we prove that it optimizes the performance of the combination, in a sense precisely defined later. Although the simple and practical combination principle has long been known in a number of fields, e.g., in artificial intelligence and computational learning theory (see, e.g., [3], from which we use mathematical results for the proof of optimality), nevertheless, to the best of our knowledge, it is fundamentally new in the context of MAC protocols.

A. The Basic Model

As Fig. 2 shows, at any given node, M MAC protocols $P_1, \dots, P_M$ have been selected to be combined.[2] The final decision whether or not to transmit at a given time is reached by appropriately combining the proposed decisions of the M component protocols.

Each protocol P_i runs locally and in each slot t produces a decision $D_{i,t}$, $1 \le i \le M$, where $D_{i,t} = 1$ is interpreted to mean that P_i would transmit in slot t and $D_{i,t} = 0$ is interpreted to mean that P_i would not transmit in slot t. We also allow intermediate values $0 < D_{i,t} < 1$ that are interpreted as probabilities, to account for protocols that use randomization. For example, $D_{i,t} = 0.7$ means that P_i would transmit with probability 0.7 in slot t. No assumptions about how each component protocol reaches its decision are made—this is completely arbitrary.

The meta-protocol is an algorithm that runs *locally* at each node and combines the component decisions $D_{i,t}$, $1 \le i \le M$, to produce a combined result D_t which is again a number in [0, 1], with the same interpretation as $D_{i,t}$. The final *binary* decision $\tilde{D}_t \in \{0, 1\}$ is derived from D_t by drawing a random binary value that takes the value 1 with probability D_t and the value 0 with probability $1 - D_t$.

Remark: We could round D_t deterministically to 0 or 1, but this would result in poorer performance when the value happens to fall often around the middle of the interval [0, 1]. For example, if $D_t = 0.51$ holds over a long sequence of slots, then deterministic rounding would result in transmission in *every* slot, excluding success if there is a conflicting node that also wants to transmit. On the other hand, in this example, random rounding generates transmissions in about 51% of the slots randomly, still allowing a chance for success.

The value of D_t is computed as a function of the weighted average of the $D_{i,t}$ values:

$$D_t = F\left(\frac{\sum_{i=1}^{M} w_{i,t}\, D_{i,t}}{\sum_{i=1}^{M} w_{i,t}}\right). \tag{1}$$

The function F can be chosen in several ways. One simple choice is $F(x) = x$, that is, D_t is made equal to the weighted average of the $D_{i,t}$. Another choice is a step function, which rounds the result to 0 or 1 depending on whether the weighted average is below 0.5 or not. It turns out, as shown in the next subsection, that optimality is achieved by a function that is in between these two choices. This function linearly grows from 0 to 1 in an interval $[(1/2) - c, (1/2) + c]$ and it is truncated to 0

[2] M is unrelated to the number of network nodes.

and 1 before and after the interval, respectively. Formally, $F(x)$ is defined as

$$F(x) = \begin{cases} 0 & \text{if } x < \frac{1}{2} - c \\ \frac{1}{2c}\left(x - \frac{1}{2} + c\right) & \text{if } \frac{1}{2} - c \le x \le \frac{1}{2} + c \\ 1 & \text{if } x > \frac{1}{2} + c. \end{cases} \quad (2)$$

The parameter c depends on another parameter $\eta > 0$ that controls the update of the weights. The following dependence makes it possible to prove the optimality: $c = (1/2) \cdot [(1 + e^{-\eta})/(1 - e^{-\eta})] \cdot \ln[2/(1 + e^{-\eta})]y$.

The meta-MAC protocol maintains the weights used in (1). The positive number $w_{i,\,t}$ is the weight of protocol P_i for slot t. At the end of each slot, the weights are updated using the local network feedback.

Let us remark that, in general, the feedback that is available for a MAC protocol can have a serious impact on the performance. For example, in slotted Aloha type protocols, a common feedback model is the *ternary feedback* that allows the user to know whether a successful transmission occurred, a collision occurred, or the channel remained idle in the slot. An interesting consequence of this is that for a fully connected network with infinite user population, no such protocol can achieve an average throughput of more than 0.568 packets/slot, no matter what kind of backoff mechanism is used [15]. In contrast to this, with more refined feedback, consisting of the exact number of users that transmitted in the slot, the throughput can be brought arbitrarily close to 1 packet/slot, which is the theoretical limit for the fully connected case [13].

In our model, in order to maintain general applicability, we do not want to restrict ourselves to a specific type of feedback. Rather, we only assume that enough information is available from which the meta-protocol can conclude (or estimate) *at the end of the slot* whether the decision for the slot was right or wrong. We call this *correctness feedback*, and it can be realistically obtained in many cases. For example, from the ternary feedback, we can easily conclude whether the decision was correct or not: if we decided to transmit and it was successful, then the decision was right; on the other hand, if a collision occurred, then it was a wrong decision. If there was a packet in the queue but we decided not to transmit, then if the channel remained idle, that implies the decision was wrong, since the slot was wasted. If the channel did not remain idle, i.e., it was used by at least one other node, then it was a right decision not to transmit. If the queue was empty, then refraining from transmission was, of course, right. We do not restrict ourselves regarding how this correctness feedback is achieved, i.e., from what actual data it is computed.

Having the above explained correctness feedback, the weight update algorithm works as follows. Let y_t denote the feedback:

$$y_t = \begin{cases} 1 & \text{if the decision in slot } t \text{ was correct} \\ 0 & \text{if the decision in slot } t \text{ was incorrect.} \end{cases}$$

Then the correct decision z_t can be retrospectively computed as $z_t = \tilde{D}_t y_t + (1 - \tilde{D}_t)(1 - y_t)$, that is, $z_t = \tilde{D}_t$ if $y_t = 1$, otherwise $z_t = 1 - \tilde{D}_t$. Of course we cannot simply set the decision for slot t according to z_t, since z_t only becomes known at the end of the slot. Using z_t, the weights are updated according to the following simple exponential rule:

$$w_{i,\,t+1} = w_{i,\,t} \cdot e^{-\eta|D_{i,\,t} - z_t|}. \quad (3)$$

This weight update rule has an appealing interpretation. The term $|D_{i,\,t} - z_t|$ in the exponent represents the deviation of protocol i from the correct decision. If this deviation is zero, then the weight of P_i remains unchanged. Otherwise it decreases the weight of P_i such that with increasing deviation (i.e., errors) the decrement grows. This means, due to the normalization in (1), that after each slot the relative weight of those protocols that made a correct decision will grow, while those which made a mistake will lose relative weight. In this way, the weights essentially reflect the "credit history" of the component protocols. The constant $\eta > 0$ controls how fast the weights can change.

Note that the direct use of (3) can cause underflow in the number representation, since the weights decrease exponentially, but never grow. This problem is easily solved in practice, by renormalizing the weights after each update. One can also set a minimum value below which no weight can drop. Renormalization does not change the *relative* sizes of the weights, and since they are only used in a normalized way in computing the combined value D_t by (1), therefore, only their relative sizes matter.

Having introduced the needed concepts, we now summarize our meta-MAC protocol.

RWM Meta-MAC Protocol:

Initialization: set all weights to 1.

In slot t do:

- *At the beginning of the slot*:

  Compute the component decisions $D_{1,\,t}, \ldots, D_{M,\,t}$.

  If there is no packet in the queue, then set $\tilde{D}_t = 0$ else compute

  $$D_t = F\left(\frac{\sum_{i=1}^{M} w_{i,\,t} D_{i,\,t}}{\sum_{i=1}^{M} w_{i,\,t}}\right).$$

  Randomly round D_t to the final binary decision $\tilde{D}_t$, according to $\Pr(\tilde{D}_t = 1) = D_t$ and $\Pr(\tilde{D}_t = 0) = 1 - D_t$.

  If $\tilde{D}_t = 1$, then transmit the first packet in the queue, otherwise refrain from transmission.

- *At the end of the slot*: Using the decision $\tilde{D}_t$ and the feedback y_t compute $z_t = \tilde{D}_t y_t + (1 - \tilde{D}_t)(1 - y_t)$ and update all weights according to $w_{i,\,t+1} = w_{i,\,t} \cdot e^{-\eta|D_{i,\,t} - z_t|}$.

B. Optimality

Now, having defined the meta-MAC protocol, we prove that it achieves optimal performance, in a well-defined sense, among all other possible combinations of the same component protocols.

A key question is how we compare the performance of different combinations of the same component protocols. The first thing that may come to mind is to compare the average throughput per slot. Doing this directly, however, is not realistic, since without any further restriction a *fully arbitrary* meta-protocol may use another "built in" MAC protocol and this may achieve better throughput than any of the component protocols, so finally the meta-protocol may end up not using the component decisions at all. In this way, we would have to assess the performance of *all possible* MAC protocols, which finally would not tell anything about how good is the *combination*. To avoid such problems and to guarantee that we really compare the possible *combinations* of the same component protocols, using the same feedback, we carefully develop a formal framework for the comparison.

In each slot t we measure the loss of the meta-protocol by the probability that the decision is incorrect, which is $\Pr(\tilde{D}_t \neq z_t)$. The way we defined $\tilde{D}_t$, D_t, and z_t implies that the loss probability is equal to the absolute error between D_t and z_t, that is, $\Pr(\tilde{D}_t \neq z_t) = |D_t - z_t|$. Now let $t = 1, \ldots, T$ be a sequence of slots. The per slot loss of the protocol over this sequence is measured by $L = (\sum_{t=1}^{T} |D_t - z_t|)/T$. By $\Pr(\tilde{D}_t \neq z_t) = |D_t - z_t|$, this is the probability of incorrect decision, averaged over the slots. In other words, we can use L as the (average) probability of wrong decision. For short, we refer to this quantity as the *loss* of the protocol. To emphasize that this loss depends on the feedback sequence $\boldsymbol{y} = (y_1, \ldots, y_T)$, we use the notation $L(\boldsymbol{y})$. Similarly, we can define the loss of each component protocol P_i by using the same formulas but distinguished by the index i:

$$L_i(\boldsymbol{y}) = \frac{\sum_{t=1}^{T} |D_{i,t} - z_t|}{T}.$$

Now we can measure the quality of the *combination* by measuring how much loss is due purely to the meta-protocol. This is obtained by comparing the loss of the combined protocol to the loss of the best component protocol. To this end, we define the *combination loss* by $C(\boldsymbol{y}) = L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y})$. The meaning of this is even clearer if, by rearranging, we write $L(\boldsymbol{y}) = \min_i L_i(\boldsymbol{y}) + C(\boldsymbol{y})$ which shows that the combination loss is really the additional loss incurred over the best component protocol, due to the combination. If $C(\boldsymbol{y}) < 0$, then the combination results in actual improvement over the best component protocol. If $C(\boldsymbol{y}) = 0$, or $C(\boldsymbol{y}) > 0$, but small, then we do not improve over the best component, but at least achieve or approximate its performance. This is still remarkable, since we do not know in advance which is the best protocol.

We capture the concept of optimal combination by looking for the meta-protocol that minimizes the combination loss. This is quite natural, since once the component protocols are given, the value of $\min_i L_i(\boldsymbol{y})$ is a constant for a given $\boldsymbol{y}$, so minimum overall loss is achieved for the given $\boldsymbol{y}$ if the combination loss is minimized.

Of course, all the above values depend on the feedback sequence $\boldsymbol{y}$ that represents the behavior of the rest of the network. Since we neither know this sequence in advance nor do we have any *a priori* probability distribution over the possible feedback sequences, therefore, we measure the performance of the meta-protocol by the *worst-case combination loss*, defined as $W = \max_{\boldsymbol{y}} C(\boldsymbol{y}) = \max_{\boldsymbol{y}}\{L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y})\}$ where the maximum is taken over all possible feedback sequences. Thus, W is a *guaranteed* upper bound on the actual combination loss for *any* feedback sequence. In this way, it characterizes how good the combination is, and it is natural to look for a protocol that can lower W to the smallest possible value.

First, we show that the worst-case combination loss can never be negative for any combination protocol, which is not obvious from the definition. This implies that the ideal goal can only be to achieve $W = 0$. Second, we prove that our RWM meta-MAC protocol in fact achieves the ideal $W = 0$ value asymptotically, so in this sense it is asymptotically optimal among *all* other possible protocols that combine the same components with the same feedback.

Theorem 1: Let P be any protocol that combines the component protocols $P_1, \ldots, P_M$, in the same framework as RWM, but may arrive at the decision in an arbitrary different way. Let the worst-case combination loss of P be W_P. Then $W_P \geq 0$ always holds. Moreover, the RWM protocol is optimal in that it asymptotically achieves the optimal 0 lower bound in the following sense: for any $\epsilon > 0$ and for any sufficiently large time horizon T there exists a choice of the parameter η such that for any feedback sequence $\boldsymbol{y}$ of length T the combination loss $C(\boldsymbol{y})$ of RWM is less than ϵ. Specifically, if $T > [(a/2\epsilon) + \sqrt{(a^2/\epsilon^2) + (b/\epsilon)}]^2$ where $a = \sqrt{\ln(M+1)/2}$ and $b = 2\log_2(M+1)$, then $C(\boldsymbol{y}) < \epsilon$ holds for any feedback sequence $\boldsymbol{y}$ of length T if the appropriate η is used. Consequently, $0 \leq W_{\text{RWM}} < \epsilon$ also holds, where W_{RWM} is the worst-case combination loss of the RWM meta-MAC protocol.

Proof: See the Appendix.

A few comments are pertinent here concerning the statement of Theorem 1. In the worst case, no combination protocol in the considered setting can outperform the best component protocol. Of course, we do not know in advance which component protocol is best for the actual situation, so we cannot simply run that protocol and ignore the rest. The RWM protocol asymptotically achieves this best possible performance, i.e., the resulting loss automatically approaches the loss of the component protocol that is the best for the actual sequence, even though neither the sequence nor the identity of the best component protocol is known in advance. Moreover, the asymptotic optimality of RWM is achieved without fully using the history of earlier decision and feedback values. The history is used only in a simple and very "condensed" way, via the weights $w_{i,t}$. The decision is also made by a simple rule. Although the theorem allows arbitrarily complicated algorithms for P, it is interesting to note that additional complexity cannot further improve the quality in the stated sense.

C. Fairness, Stability, and Convergence

Fairness, both short term and long term, are important properties of a MAC protocol. In some sense, the meta-MAC protocol "inherits" the fairness properties of its components, and, in particular, the current best protocol for the network conditions. The meta-protocol has its own fairness properties and the characterization of its fairness remains to be completed.

In the experiments that follow, we only considered a network load that was uniformly distributed over the entire network. In order to understand the stability of the combination principle and how fast the method converges, we need to consider nonuniform dynamic load conditions in our simulations. While it is clear that the larger the value of η the more rapidly the meta-protocol adapts to the feedback, it is still not well understood how η will interact under nonuniform load, noisy channels, and other similar network characteristics.

IV. Examples of the Meta-MAC Protocol

A. Examples of Protocol Combination

1) Combining Slotted Aloha and TDMA in a LAN: To see how our meta-MAC protocol combines the advantages of two complementary approaches in LANs, we selected a slotted Aloha protocol and a TDMA protocol, and compare our results to IEEE 802.3 [14].

Using a discrete event simulator, we modeled a 10 Mbits/s LAN with $N = 100$ nodes. Network traffic was introduced according to a Poisson arrival process with a mean of λ packets per slot uniformly distributed among the nodes, with fixed length packets of 100 and 500 bytes. The meta-protocol used the value of $\eta = 1$ in updating the weights at the end of a slot, and perfect channel feedback was assumed. Each data point represents an average of many simulation runs, each of which simulated 600 s real time achieving a confidence interval of over 90%.

The slotted Aloha protocol we used is stable [8]: if a given slot is idle (has a collision) then the node multiplies (divides) its transmission probability by a constant q (we used $q = 2$). The main difference between this backoff mechanism and binary exponential backoff (BEB) is that the backoff interval is not reset to one on a successful transmission. In each slot, the protocol returns a decision representing its transmission probability.

We used a simple TDMA protocol with a frame length of $N = 100$. Each node has a unique identifier i, $1 \leq i \leq N$, which is used as its assigned slot in the frame. For node i, the protocol returns a binary decision depending on whether or not it is i's assigned transmission slot.

A main difference between IEEE 802.3, the component protocols and the resulting meta-protocol is that 802.3 uses an asynchronous approach with collision detection to transmit variable length packets. Since the packet length for the meta-protocol is fixed, we simplified our simulation of 802.3 to use fixed size packets too. Our simulation results for meta-MAC, TDMA, and slotted Aloha did not differ significantly for 100 and 500 bytes; thus, for those protocols, we only show results for one packet size.

For the considered protocols, we measured the average number of successful packet transmissions per second (throughput or, equivalently, channel utilization), and the average time necessary to successfully access the channel (the access delay).

Fig. 3 shows that at low loads the throughput of slotted Aloha, TDMA, and the meta-protocol are all the same, almost matching the arrival rate. As the load increases, the meta-protocol tracks the throughput of TDMA consistently. Although 802.3 follows the meta-MAC curve at low and middle loads, at high loads it drops behind and remains constant.[3]

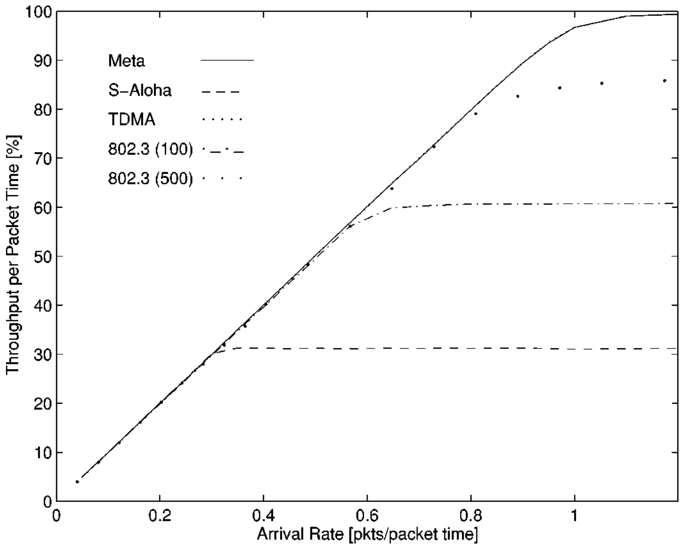


Fig. 3. Throughput, as a function of network load.

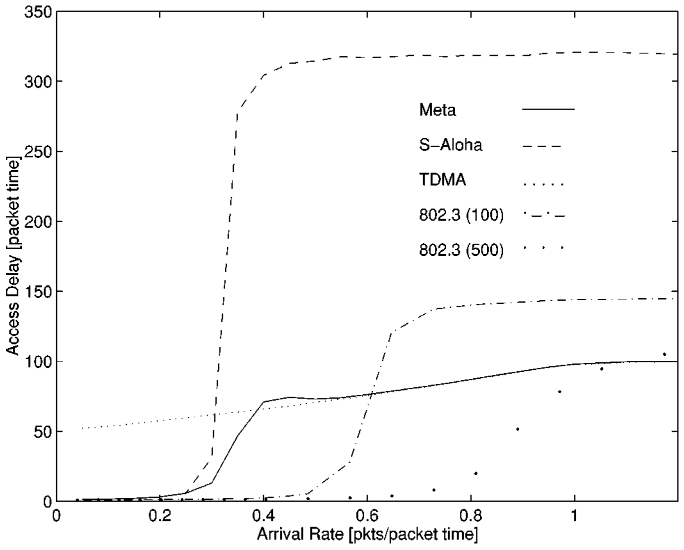


Fig. 4. Access delay, as a function of network load.

Fig. 4 shows that at low load, the delay of the meta-protocol corresponds to the delay of slotted Aloha; and as load increases, its delay corresponds to that of TDMA. The small "overshoot" in Figs. 4 and 5 for the meta-protocol occurs at the network load at which the weights of the component protocols in the meta-protocol are switching from slotted Aloha to TDMA. Fig. 6 captures how the meta-protocol shifts its reliance on each protocol as the load conditions in the LAN change. As can be observed, 802.3 outperforms the meta-protocol at the middle loads. At low and high loads, our meta-MAC has similar delay characteristics or even outperforms 802.3.

Thus, when combining protocols of different types, this example shows that the meta-protocol is able to automatically adjust to the best protocol for the current network conditions. We

[3]802.3 does not break down since the number of nodes is 100 and the minimum backoff probability is 2^{10}.

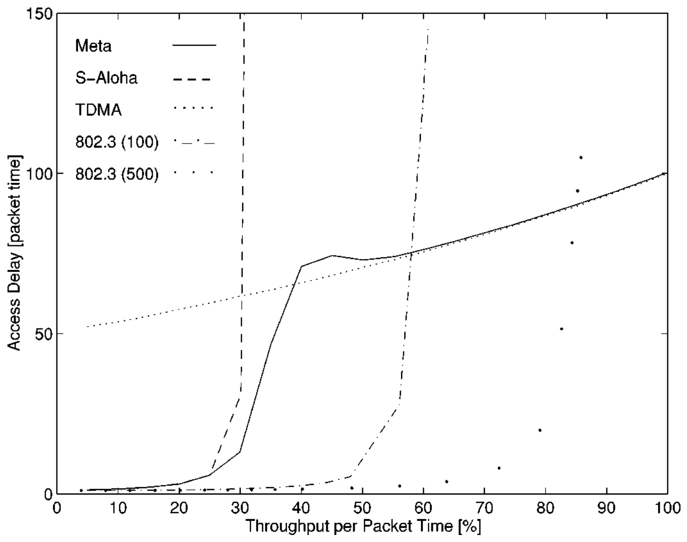


Fig. 5. Throughput-delay characteristics.

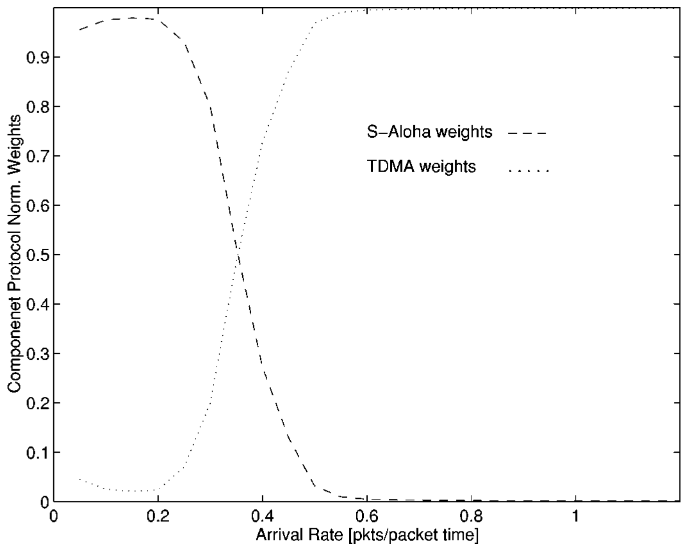


Fig. 6. Normalized average weights.

also see that under certain circumstances, the meta-MAC protocol can outperform a CSMA/CD based protocol even without the powerful features of carrier sensing and collision detection.

2) Combining Slotted Aloha and TDMA in a Mobile Multihop Network: In this subsection, we adapt the meta-protocol from the previous section to operate in a mobile multihop wireless network, and compare its performance to that of IEEE 802.11 [18]. Moreover, we also consider the effects of noisy feedback.

Obtaining channel feedback in a mobile multihop network is more difficult than in a LAN. Because of hidden terminals, the outcome is not always clear. As well, collision detection cannot be used in a wireless environment since a wireless node cannot transmit and receive simultaneously. Consequently, we enlarged the slot to accommodate a 32-byte acknowledgment packet to provide explicit feedback after a packet reception. The absence of an acknowledgment is interpreted as a collision.

The IEEE 802.11 protocol is a pure contention protocol in which the nodes directly compete for channel access using a

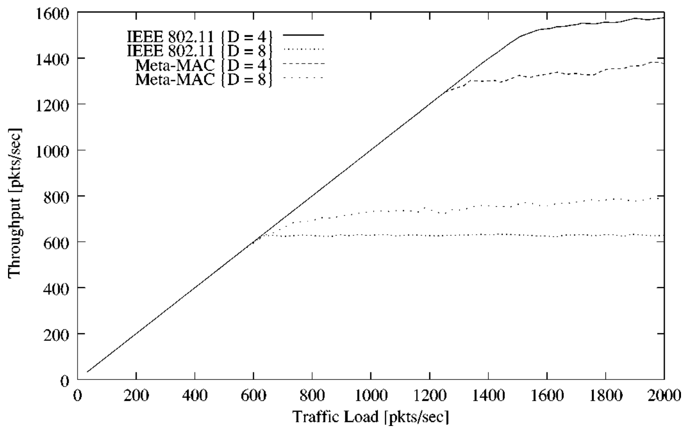


Fig. 7. Throughput, as a function of network load (perfect channel, average node degree of $D = 4$ and 8).

combination of carrier sensing and collision-avoidance handshakes. While the protocol has two methods of determining channel access rights, we only implement the *distributed coordination function* because of the distributed nature of mobile multihop networks.

Using a discrete event simulator, we modeled a mobile multihop network consisting of $N = 32$ nodes operating in a two-dimensional plane. Each simulated node was equipped with a wireless radio device capable of transmitting at a data rate of 10 Mbits/s to a distance of 300 m. For simplification, all communication was assumed to have taken place on a single channel, and a free-space propagation model was employed without capture.

Node movement was simulated using a random graph model in which the network connectivity was represented as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of wireless links. Two nodes i and j are neighbors (i.e., can directly communicate with one another) if there is an edge $(i, j) \in E$. This model was used because we were able to create a connected topology with controlled node density. Movement was simulated by creating a new random graph every 2 s. While each node does not have perfect knowledge of the topology, it does have perfect knowledge of its neighbors.

Network traffic was generated according to a Poisson arrival process with a mean of λ packets per second, and uniformly distributed among the operating nodes. Each packet contained a payload of 2048 bytes of data, and was addressed to a random neighbor.

Due to space limitations, we are only able to show results for low node degrees. Each data point represents an average of several simulation runs, each of which lasted 300 s real time, resulting in a confidence interval of over 90%. Figs. 7 and 8 show the performance of the meta-MAC and 802.11 protocols using a perfect channel model. Figs. 9 and 10 show the performance of both protocols using a channel model which accommodates a bit error rate parameter of $\beta = 10^{-6}$, while Figs. 11 and 12 compare the throughput delay characteristics of the perfect and noisy channel.

As expected, the meta-protocol outperforms pure TDMA (which has a maximum throughput of 610 packets/s and a maximum average access delay of 0.052 s) with low traffic

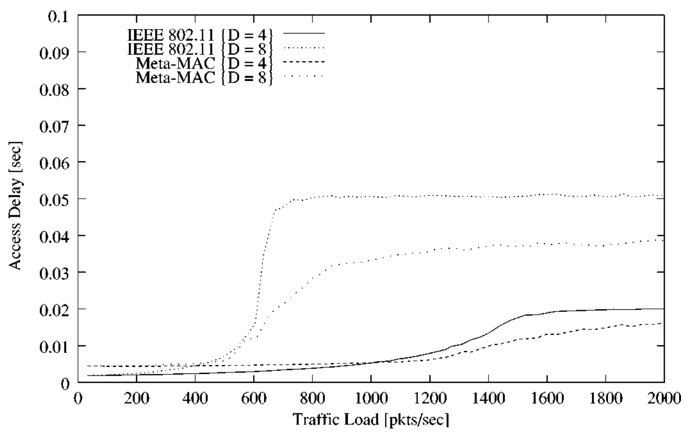


Fig. 8. Access delay, as a function of network load (perfect channel, average node degree of $D = 4$ and 8).

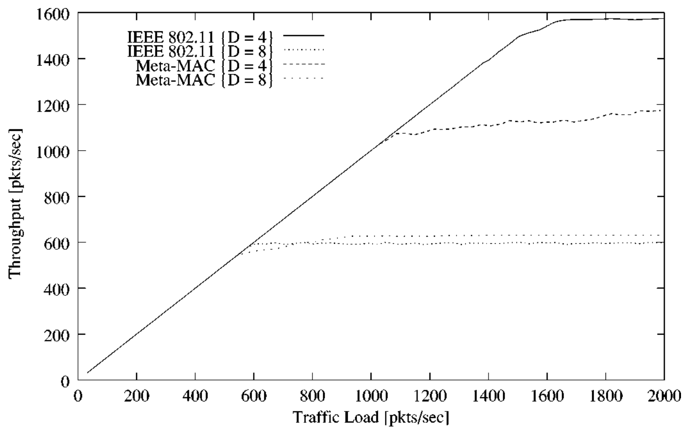


Fig. 9. Throughput, as a function of network load (noisy channel, average node degree of $D = 4$ and 8).

load or nodal degree (see Figs. 7 and 8). When the average nodal degree is greater than 4, the throughput performance of the meta-protocol is better than that of the IEEE 802.11 protocol due to the control packet overhead and time needed for collision resolution. On the other hand, the access delay of the meta-protocol is always greater than that of IEEE 802.11 because the meta-protocol must wait until the beginning of a slot to send a packet.[4] Thus, the performance gap between the meta-protocol and the IEEE 802.11 protocol will be reduced in a more realistic simulation environment.

Comparing Figs. 7 and 8 to Figs. 9 and 10, we can clearly see that the presence of noise in the channel has a much greater impact on the performance of the meta-protocol than that of IEEE 802.11. This is especially true at the lower average nodal degrees when good channel feedback is critical to the success of the slotted Aloha component protocol. Since there is an increased number of lost packets and acknowledgments, the slotted Aloha component protocol has degraded performance. The effect of the noisy channel is mitigated when the nodal degree is higher since the TDMA component protocol only returns a positive transmission decision once per frame.

[4]Our simulations did not take into account the additional overhead needed to achieve a synchronous protocol.

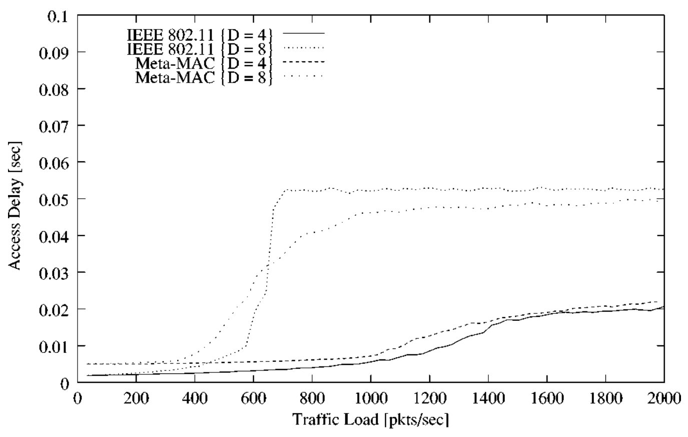


Fig. 10. Access delay, as a function of network load (noisy channel, average node degree of $D = 4$ and 8).

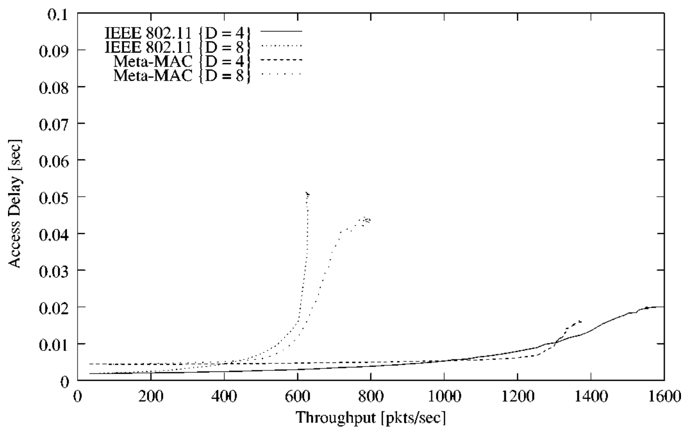


Fig. 11. Throughput-delay characteristics for perfect channel (average node degree of $D = 4$, 8).

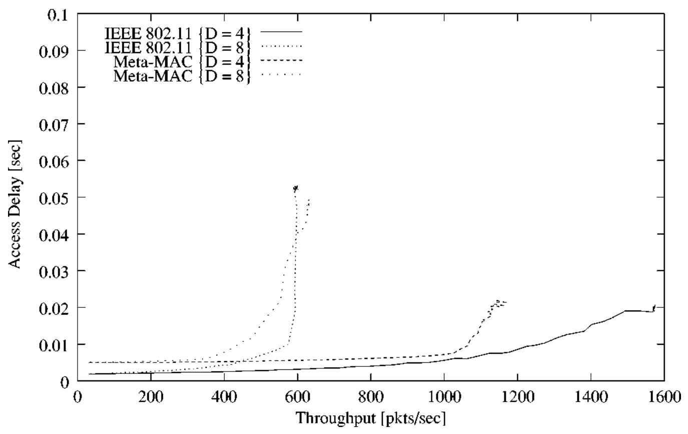


Fig. 12. Throughput-delay characteristics for noisy channel (average node degree of $D = 4$, 8).

B. Examples of Parameter Optimization

While the meta-protocol can combine arbitrary different protocols, it can also combine the same protocol but using different parameters, e.g., TDMA protocols with different frame lengths, or p-persistent slotted Aloha protocols with different values of p, etc. In this way, our aggregation approach provides a way to

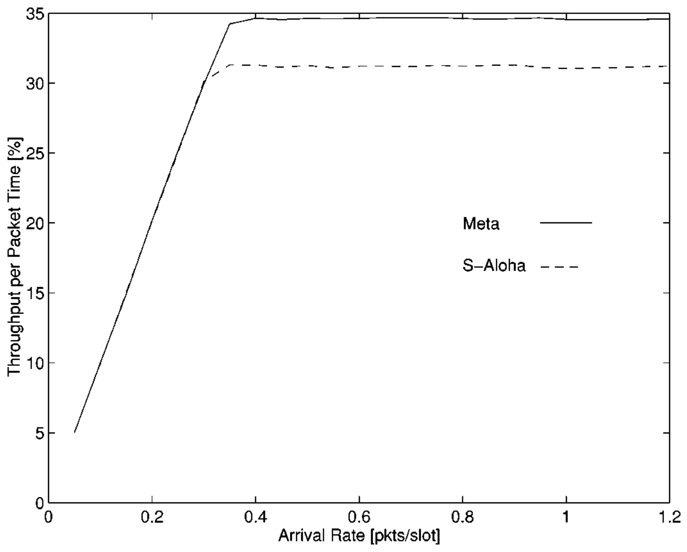


Fig. 13. Throughput, when optimizing p-persistent slotted Aloha in a LAN.

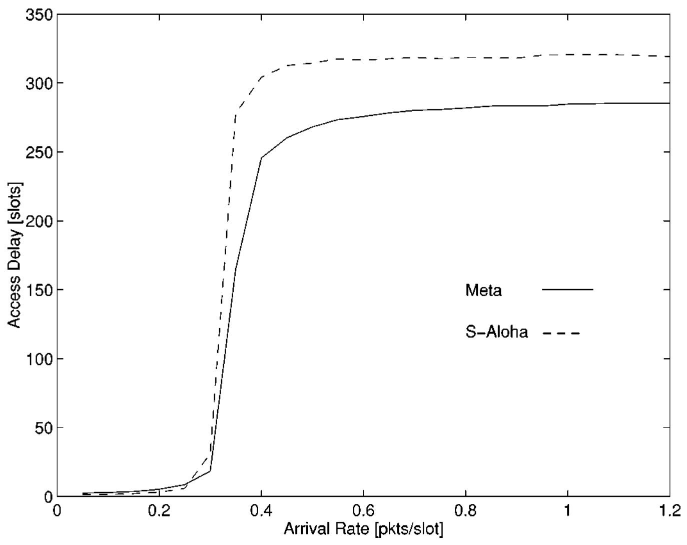


Fig. 14. Access delay, when optimizing p-persistent slotted Aloha in a LAN.

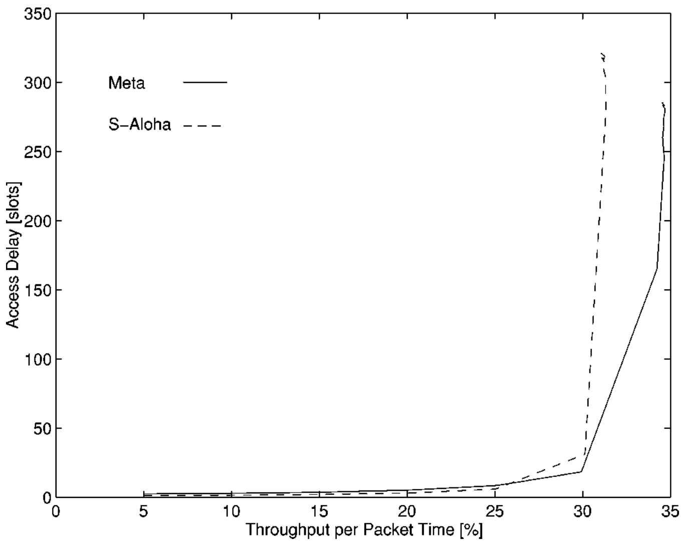


Fig. 15. Throughput-delay characteristics when optimizing p-persistent slotted Aloha in a LAN.

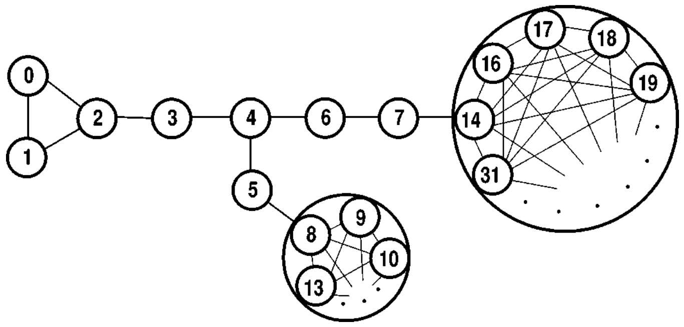


Fig. 16. Static multihop network topology for the TDMA based meta-protocol.

automatically optimize critical protocol parameters. In this subsection we investigate the application of the combination principle to parameter optimization.

1) Optimizing p in p-Persistent Slotted Aloha in LANs: In this example, we combine p-persistent slotted Aloha protocols, each of which differs only in its transmission probability p. A p-persistent protocol does not have a backoff mechanism; it relies solely on its transmission probability p to decide in which slot to transmit next.

At high load, when all nodes always have packets to send, the probability of a successful transmission is given by $p \cdot (1 - p)^{N-1}$. The value of p for optimal throughput is $p = 1/N$, i.e., the number M of component protocols should be $M > \log(N)$. For our $N = 100$ node LAN with the same simulation parameters as in Section IV-A-1, we combine $M = 15$ p-persistent Aloha protocols where protocol P_j has transmission probability $p_j = 2^{-j}$, $j = 1, \ldots, M$. At each node i, each protocol P_j, $j = 1, \ldots, M$, returns a decision $D_{j,t}$, $0 \leq D_{j,t} = p_j \leq 1$, representing its transmission probability.

Figs. 13–15 show the results. What is interesting is that the meta-protocol outperforms the stable slotted Aloha protocol. The reason is that the transmission probability of the stable slotted Aloha protocol is limited to powers of 1/2 for its transmission probability. Specifically, this protocol "jumps" between probabilities $1/2^k$ and $1/2^{k+1}$ that bracket the optimal transmission probability, p_{opt}, $1/2^k \leq p_{opt} \leq 1/2^{k+1}$ for the current slot. In contrast, the meta-protocol can actually converge to p_{opt}, which gives rise to its improved performance.

Thus the meta protocol dynamically adjusts its transmission probability automatically according to the network conditions.

2) Optimizing TDMA Schedules in a Static Multihop Network: We give another example of the combination principle for parameter optimization in a static (i.e., not mobile) multihop network combining protocols of the same type with different parameters. A discrete event simulator was used to implement the meta-protocol built on a static multihop network with $N = 32$ nodes. For our experiment, Fig. 16 shows the static network used. Here, the small circles represent nodes (with the given node identifier) and the lines connecting the circles represent bidirectional wireless links. The large circles represent fully connected subnetworks of size 6 and 18, involving nodes 8–13 and 14–31, respectively.

TABLE I
FRAME LENGTH AND SLOT ASSIGNMENTS MADE BY THE META-MAC PROTOCOL

Node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Frame Size	4	4	4	4	4	4	4	4	8	8	8	8	8	8	16	32
Slot Assignment	2	3	0	1	2	3	0	2	5	1	2	6	4	0	3	22
Node	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Frame Size	32	32	16	32	32	16	16	16	32	16	32	32	32	16	32	32
Slot Assignment	21	17	2	19	5	13	15	7	10	14	1	9	25	11	27	26

In this experiment, we use schedules with lengths that are powers of two. This is essential since neighborhoods or neighboring nodes using different frame lengths must interwork with each other resulting in a nonconflicting schedule. Thus, the frames must be capable of being embedded in one another. To ensure that there is at least one schedule for each node, we have to take the worst-case bound of a simple global TDMA and use $m = \lceil \log_2 N \rceil$ different TDMA frame lengths: $\{T_j = 2^j\}$, where $1 \leq j \leq m$. For each such TDMA frame length T_j, there are 2^j distinct schedules, each with a transmission right in a different slot. Thus, at each node i, there are a total of $M = \sum_{j=1}^{m} 2^j = 2^{m+1} - 1$ component TDMA protocols. Initially, each node is assigned all slots from each length TDMA schedule, i.e., the weights for all $2^{m+1} - 1$ protocols are the same. The meta-protocol reaches a nonconflicting schedule if during the run at each node there is one (and only one) TDMA protocol with an assigned normalized weight of almost one. This means that all protocol weights except one must be infinitesimal. The TDMA component protocol with the normalized weight of approximately 1 will determine the frame length and assignment for the given node.

Some simple modular arithmetic on the frame size together with a slot counter t can be used to determine the decisions of the individual protocols, i.e., at each node i, each protocol P_j, returns a binary decision $D_{j,t} = 1$ or $D_{j,t} = 0$ representing whether or not the current slot t is assigned to i by P_j.

Table I shows, for each node in the network, the TDMA frame size and slot assignment in that frame size to which the meta-protocol converged. The meta-protocol converged to frame lengths that were very close to optimal with no conflicts in the slot assignment. Since there are 18 nodes in the one fully connected subnetwork, it is expected that some nodes must be assigned a frame size of 32 rather than 16.

Thus, by combining TDMA protocols with different frame sizes in a static network, the meta-protocol automatically converged to a near optimal frame size to match each node's connectivity in the static multihop network. Moreover, a nonconflicting slot assignment was made.

V. CONCLUSION

This paper presented a systematic and automatic method to combine any set of existing MAC protocols into a single higher layer, or *meta*-MAC protocol. This approach guarantees that the overall performance of the meta-MAC protocol matches the performance of the best component protocol for the current network conditions. This optimization is achieved without knowing, *a priori*, which component protocol performs the best in potentially highly dynamic network conditions. Moreover, this optimization is automatic and runs entirely locally, i.e., it requires no centralized control nor any message passing, using only local network feedback information. We have outlined the method, and proved that the resulting meta-MAC protocol achieves optimal performance. Through extensive simulation with different types of networks and component MAC protocols, we have shown the effectiveness of our combination principle in a wide variety of situations.

APPENDIX A

Proof of Theorem 1: First we show that $W_P \geq 0$ holds for any meta-protocol P satisfying the conditions of the theorem. Let P_1 be one of the component protocols. Since we consider the worst case with respect to the feedback sequence $\boldsymbol{y}$, we can choose a "malicious" feedback sequence. Since y_t and z_t are connected via $z_t = \tilde{D}_t y_t + (1 - \tilde{D}_t)(1 - y_t)$, and $\tilde{D}_t$ is already decided at the beginning of the slot, independently of the feedback y_t, then by choosing the value of y_t, we can control z_t *arbitrarily.* So, let us set y_t such that z_t takes the value

$$z_t = \begin{cases} 1 & \text{if } D_t < D_{1,t} \\ 0 & \text{if } D_t \geq D_{1,t}. \end{cases}$$

This ensures $|D_t - z_t| \geq |D_{1,t} - z_t|$, which implies $L(\boldsymbol{y}) - L_1(\boldsymbol{y}) \geq 0$ for this particular $\boldsymbol{y}$. If $L_1(\boldsymbol{y})$ is replaced by $\min_i L_i(\boldsymbol{y}) \leq L_1(\boldsymbol{y})$, then this term can only decrease, so the difference in $L(\boldsymbol{y}) - L_1(\boldsymbol{y})$ can only grow. Thus, we have $C(\boldsymbol{y}) = L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y}) \geq 0$ for this chosen $\boldsymbol{y}$. Maximizing with respect to $\boldsymbol{y}$ can again only increase the value, so we have $W_P = \max_{\boldsymbol{y}} C(\boldsymbol{y}) = \max_{\boldsymbol{y}}\{L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y})\} \geq 0$, which proves the first claim of the theorem.

Now we show that the RWM protocol asymptotically achieves the 0 lower bound, which proves its (asymptotic) optimality. This requires, however, a complicated proof, using methods that have been used in a prediction model in the context of computational learning theory by Cesa-Bianchi *et al.* [3]. To help understand the main concept, first we show how the proof works for a simplified case; and after that, we consider the general case.

Thus, let us consider first the following simplified case. Assume that each component decision $D_{i,t}$ can only be 0 or 1, that is, intermediate values are not allowed. Further, let the combined decision be simply the weighted majority of the component decisions (i.e., deterministic rounding). Let S_t denote the sum of the weights in slot t: $S_t = \sum_{i=1}^{M} w_{i,t}$. Let R_t be the set of those component protocols which made the right decision in slot t, that is, $R_t = \{i | D_{i,t} = z_t\}$ and set $E_{i,t} = |D_{i,t} - z_t|$. Thus, $E_{i,t}$ is the error indicator: since now $D_{i,t} \in \{0, 1\}$, therefore, it takes the value 1 if protocol i made the wrong decision in slot

t and 0 otherwise. Now let us consider the sum of the weights in slot $t+1$, after the update. According to our update rule, we have $S_{t+1} = \sum_{i=1}^{M} w_{i,t+1} = \sum_{i=1}^{M} w_{i,t} \cdot e^{-\eta E_{i,t}}$. This can be decomposed into two sums: one for those component protocols that made the right decision in slot t ($E_{i,t} = 0$), and one for those that made a mistake ($E_{i,t} = 1$). Using that $e^{-\eta E_{i,t}}$ becomes 1 or $e^{-\eta}$ for $E_{i,t} = 0$ or 1, respectively, we have

$$S_{t+1} = \sum_{i \in R_t} w_{i,t} + \left(\sum_{i \notin R_t} w_{i,t} \right) \mathrm{e}^{-\eta}. \tag{4}$$

Consider now the case when the meta-protocol made the wrong decision in slot t. Then at least half of the total weight had to be on the wrong side, yielding $\sum_{i \notin R_t} w_{i,t} \geq S_t/2$ and $\sum_{i \in R_t} w_{i,t} \leq S_t/2$. Since for $i \notin R_t$ the weight is multiplied by a factor less than 1, therefore, the right-hand side of (4) is bounded from above by the value when the sum for $i \notin R_t$ is the smallest possible (i.e., $S_t/2$) and then the other sum is also $S_t/2$, since it contains the rest of the weights. This may not be the actual distribution, but in this way we surely get an upper bound on the right-hand side of (4). Thus, we have that if the meta-protocol made an error in slot t, then $S_{t+1} \leq (S_t/2) + (S_t/2)\mathrm{e}^{-\eta} = S_t((1+\mathrm{e}^{-\eta})/2)$. Consider now the case when the meta-protocol makes k errors over a slot sequence of length T. Since, according to the update rule, the weights can never grow and whenever the meta-protocol errs S_t decreases at least by a factor of $(1+\mathrm{e}^{-\eta})/2$, therefore, after k errors S_t has to decrease at least by the kth power of the factor, so at the end of the slot sequence we have $S_T \leq S_0((1+\mathrm{e}^{-\eta})/2)^k$. Taking into account that the initial weight sum is $S_0 = M$, we obtain

$$S_T \leq M \left(\frac{1+\mathrm{e}^{-\eta}}{2} \right)^k. \tag{5}$$

Now let i be the index of the component protocol that performs the best for the given sequence and assume it made ℓ errors. Then, by the weight update rule, its weight was decreased ℓ times by a factor of $\mathrm{e}^{-\eta}$. Given that initially each weight is 1, we have at the end of the sequence $w_{i,T} = \mathrm{e}^{-\eta\ell}$. Since the total weight can never be smaller than any individual weight, therefore, $S_T \geq w_{i,T}$ must hold. This implies by (5) and $w_{i,T} = \mathrm{e}^{-\eta\ell}$ the following inequality: $M(1+\mathrm{e}^{-\eta})/2)^k \geq \mathrm{e}^{-\eta\ell}$. By rearranging this inequality, we obtain the following bound on the number of meta-protocol errors in terms of the errors made by the best component protocol:

$$k \leq \frac{\eta}{\ln 2 - \ln(1+\mathrm{e}^{-\eta})} \ell + \frac{\ln M}{\ln 2 - \ln(1+\mathrm{e}^{-\eta})}. \tag{6}$$

Now one can show with lengthy calculations (see [3]) that if η is chosen as $\eta = -\ln g(T/\ln M)$, where $g(x) = 1 - 2(\sqrt{1+x} - 1)/x$, then the estimation $k - \ell \leq \sqrt{T \ln M} + (\log_2 M/2)$ can be obtained. Dividing both sides by T and taking into account that $k/T = L(\boldsymbol{y})$, $\ell/T = L_i(\boldsymbol{y}) = \min_j L_j(\boldsymbol{y})$, we have $L(\boldsymbol{y}) - \min_j L_j(\boldsymbol{y}) \leq \sqrt{(\ln M/T)} + (\log_2 M/2T)$. Since the derivation of this bound did not depend on the choice of the feedback sequence $\boldsymbol{y}$, then it should hold for *any* such sequence, yielding $C(\boldsymbol{y}) = L(\boldsymbol{y}) - \min_j L_j(\boldsymbol{y}) \leq \sqrt{(\ln M/T)} + (\log_2 M/2T)$, which implies $W = \max_{\boldsymbol{y}}(L(\boldsymbol{y}) - \min_j L_j(\boldsymbol{y})) \leq \sqrt{(\ln M/T)} + (\log_2 M/2T)$. Given that M is a constant, these inequalities imply that for any $\boldsymbol{y}$, $C(\boldsymbol{y}) \to 0$ and, consequently, $W \to 0$ holds as $T \to \infty$.

Now recall that this was a simplified case to show the principle of the proof. This simplified case does not cover the actual algorithm, since we excluded intermediate values for $D_{i,t}$, ignored the randomization (thus allowed only 0-1 valued loss) did not use the function (2) as well as the relationship of the parameter c of the function with η. These are needed for the general proof, which is substantially more complex. Fortunately, however, we can use the results of [3], since our model can be exactly mapped into their model. In [3] a prediction model is considered in which M "experts" predict a sequence of events with binary outcomes over time and these predictions are combined into a final prediction via a weighted combination, where the weights are updated the same way as in the meta-MAC protocol. (Variants of such prediction models are used, for example, to capture the situation when an investor wants to predict stock market trends, using expert advice.) Our meta-MAC protocol can be directly and exactly mapped in the prediction model, by identifying the component protocols with the experts and the feedback sequence with the outcome sequence in the prediction model, allocating one slot for each event. The prediction model allows a class of functions for use in the combination formula (1), including our piecewise linear, sigmoid-type function (2). The prediction model uses a parameter β in the algorithm, which can be directly mapped into our η parameter by $\beta = e^{-\eta}$. In this way we can directly apply the following theorem from [3] for the general case (reformulated with our notation): Let $\boldsymbol{y} = (y_1, \ldots, y_T)$ be any sequence of outcomes of events. Further, let $\Sigma(\boldsymbol{y})$ be the accumulated absolute error of the combined prediction. Similarly, let $\Sigma_i(\boldsymbol{y})$, $i = 1, \ldots, M$, be the accumulated absolute error of the ith expert. Then $\beta(= e^{-\eta})$ can be chosen such that $\Sigma(\boldsymbol{y}) - \min_i \Sigma_i(\boldsymbol{y}) \leq \sqrt{((T\ln(M+1))/2)} + ((\log_2(M+1))/2)$ holds.

Using the above result, via the mapping between the prediction model and the meta-MAC protocol model, we obtain

$$\begin{aligned} C(\boldsymbol{y}) &= L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y}) \\ &= \frac{\Sigma(\boldsymbol{y})}{T} - \min_i \frac{\Sigma_i(\boldsymbol{y})}{T} \\ &\leq \sqrt{\frac{\ln(M+1)}{2T}} + \frac{\log_2(M+1)}{2T}. \end{aligned}$$

As this holds for any $\boldsymbol{y}$, it also should hold for the one that maximizes $C(\boldsymbol{y}) = L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y})$, yielding

$$\max_{\boldsymbol{y}}\{L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y})\} \leq \sqrt{\frac{\ln(M+1)}{2T}} + \frac{\log_2(M+1)}{2T}.$$

From this we obtain, via solving the inequality $\sqrt{((\ln(M+1))/2T)} + ((\log_2(M+1))/2T) < \epsilon$ for T, that $C(\boldsymbol{y}) \leq W_{\mathrm{RWM}} = \max_{\boldsymbol{y}}\{L(\boldsymbol{y}) - \min_i L_i(\boldsymbol{y})\} < \epsilon$ holds, whenever

$$T > \left(\frac{a}{2\epsilon} + \sqrt{\frac{a^2}{\epsilon^2} + \frac{b}{\epsilon}} \right)^2$$

where $a = \sqrt{\ln(M+1)/2}$ and $b = 2\log_2(M+1)$.

References

[1] D. Aldous, “Ultimate stability of exponential backoff protocol for acknowledgment based transmission control of random access communication channels,” *IEEE Trans. Inform. Theory*, vol. 33, pp. 219–223, 1987.

[2] R. E. Ahmea, “An adaptive multiple access protocol for broadcast channels,” in *Proc. IEEE Int. Conf. Performance, Computing Commun.*, 1997, pp. 371–377.

[3] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth, “How to use expert advice,” in *Proc. 25th Annu. ACM Symp. Theory Computing (STOC’93)*, San Diego, CA, May 1993, pp. 382–391. (Univ. Calif. Computer Res. Lab., Santa Cruz, CA, Tech. Rep. UCSC-CRL-94-33, 1994.).

[4] I. Chlamtac, A. Faragó, and H. Zhang, “Time-spread multiple-access (TSMA) protocols for multihop networks,” *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 804–812, 1997.

[5] I. Chlamtac, A. Faragó, A. D. Myers, V. R. Syrotiuk, and G. Záruba, “ADAPT: A dynamically self-adjusting media access control protocol for *ad hoc* networks,” in *Proc. IEEE Globecom General Conf.*, Rio de Janiero, Brazil, Dec. 1999, pp. 11–15.

[6] I. Chlamtac and S. Pinter, “Distributed node organization algorithm for channel access in multihop packet radio networks,” *IEEE Trans. Computers*, vol. 36, pp. 728–730, 1987.

[7] J. Håastad, F. T. Leighton, and B. Rogoff, “Analysis of backoff protocols for multiple access channels,” in *Proc. ACM Symp. Theory of Computing (STOC’87)*, New York, NY, May 1987, pp. 241–253.

[8] D. G. Jeong and W. S. Jeon, “Performance of an exponential backoff scheme for slotted-ALOHA protocol in local wireless environment,” *IEEE Trans. Veh. Technol.*, vol. 44, pp. 470–479, 1995.

[9] J. O. Limb, “Load-controlled scheduling of traffic on high-speed metropolitan area networks,” *IEEE Trans. Commun.*, vol. 37, pp. 1144–1150, 1989.

[10] F. P. Kelly, “Stochastic models of computer communication systems,” *J. Roy. Stat. Soc. (B)*, vol. 47, pp. 379–395, 1985.

[11] E. Kopsahilis, G. Papadopoulos, S. Koubias, and V. Christidis, “A simulation and measured performance of a new multiple access protocol,” in *Proc. 6th Eurotech. Conf.*, vol. 2, 1991, pp. 1105–1108.

[12] G. I. Papadimitriou and D. G. Maritsas, “Self-adaptive random-access protocols for WDM passive star networks,” in *IEE Proc.-Comput. Digit. Tech.*, vol. 142, July 1995, pp. 306–312.

[13] N. Pippenger, “Bounds on the performance of protocols for a multiple access broadcast channel,” *IEEE Trans. Inform. Theory*, vol. 27, pp. 145–151, 1981.

[14] A. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[15] B. S. Tsybakov and N. B. Likhanov, “Upper bound on the capacity of a random multiple access system,” *Problemy Peredachi Informatsii*, vol. 23, no. 3, pp. 64–87, 1987.

[16] G. Vardakas, W. Kishaba, and C. L. Fullmer, “QoS networking with adaptive link control and tactical multi-channel software radios,” in *Proc. IEEE MILCOM’99*.

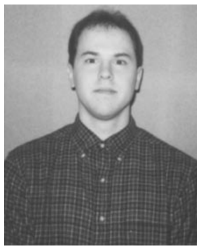
[17] C. Zhu and S. Corson, “A five-phase reservation protocol (FPRP) for mobile *ad hoc* networks,” in *Proc. IEEE INFOCOM*, 1998, pp. 322–331.

[18] *Wireless Medium Access Control and Physical Layer WG*, IEEE Draft Standard P802.11 Wireless LAN, IEEE Standards Dep., Jan. 1996.

András Faragó (M’99) received the M.S. and Ph.D. degrees in electrical engineering from the Technical University of Budapest, Hungary, in 1976 and 1981, respectively. In 1996 he received the distinguished title “Doctor of Sciences” from the Hungarian Academy of Sciences.

He joined the University of Texas at Dallas as a Professor of Computer Science in 1998. Until 1997, he was with the Department of Telecommunications and Telematics, Technical University of Budapest, where he was co-founder of the High Speed Networks Laboratory, a leading telecom research laboratory in the region. He worked as a visiting Senior Research Fellow at the University of Massachusetts at Amherst in 1991–1992. He spent a sabbatical year at Boston University in 1996. He serves as Editor for the journal *Wireless Networks*. His research focuses on algorithms, protocols, design, and analysis methods of communication networks, and has authored over 100 papers in the field.

Dr. Faragó is a member of IFIP Working Group 6.3 “Performance of Communication Systems” and was a founding member of the Hungarian Chapter of ACM.



Andrew D. Myers (S’98) received the B.S. degree (with honors) and M.S. degree in computer science from the University of Texas at Dallas, in 1997 and 1999, respectively. He is currently working towards a Ph.D. degree in computer science at the University of Texas at Dallas.

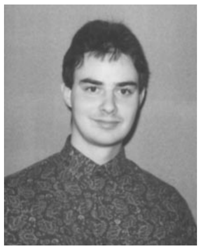
In 1997, he joined the Center for Advanced Telecommunications Systems and Services (CATSS) as a Research Assistant. From 1997 through 1999, he was involved with the DARPA sponsored Global Mobile Information (GloMo), developing and testing signaling protocols for mobile *ad hoc* networks. He is currently working in the areas of personal area networks (PANs) and mobile sensor networks.

Mr. Myers is a student member of the ACM.

Violet R. Syrotiuk (M’99) received the B.S. degree in computer science from the University of Alberta in 1983, and the M.S. and Ph.D. degrees from the University of British Columbia and the University of Waterloo in 1984 and 1992, respectively.

She is an Assistant Professor in the Department of Computer Science in the Erik Jonsson School of Engineering and Computer Science. From 1997, she was involved in a DARPA Global Mobile Information Systems project working to make the mobile environment a “first class citizen” in the defense information infrastructure. Her current research interests include medium access and network layer protocols for wireless mobile networks, network simulation, and distributed algorithms and systems.

Dr. Syrotiuk is a member of ACM, SIGMOBLE, and COMSOC.



Gergely V. Záruba (S’98) received the M.S. degree in computer engineering in 1997 from the Technical University of Budapest, Department of Telecommunications and Telematics, with excellent classification. Since 1997, he has been pursuing the Ph.D. degree.

From 1995 through 1999 he was a member of Hungary’s leading telecom research laboratory, the High Speed Networks Laboratory at the Technical University of Budapest. In 1998, he joined the Center for Advanced Telecommunications Systems and Services (CATSS) at the University of Texas at Dallas, where he is currently pursuing research on communication protocols for mobile multihop networks as a Research Assistant. His research interests also include location and tracking of mobile users and admission control in wireless networks.

Mr. Záruba is a student member of COMSOC.