Entropy-Based Estimation of Client Preferences in Wireless Push Systems

C. Liaskos, *Graduate Member, IEEE,* and G. Papadimitriou, *Senior Member, IEEE*

***Abstract*—Knowledge of the clients' preferences in communication systems enables adaptive, precise resource allocation in real-time. The present work introduces a novel scheme for estimating the clients' preferences in wireless data broadcast systems. The proposed scheme relies on the principle of maximum entropy, with statistical moments of the clients' overall waiting time serving as inputs, and the probability distribution of the broadcast data items being the output. Observations regarding periodic broadcast scheduling enabled the reduction of the convergence time well below that of related well-known approaches, in both noisy and noiseless environments. Simulation results showed that the technique can be used for accurate adaptation in minimal times, compared to the alternatives.**

***Index Terms*—Wireless networks, push, broadcasting, adaptivity.**

I. Introduction

WIRELESS data broadcasting is an efficient means of bandwidth preservation and information advertising. Exploiting the commonality in the needs of the clients, broadcasting disseminates data per user class instead of per single user, saving network resources. However, efficient broadcasting requires optimal scheduling. The data broadcasts must be serialized in a way that optimizes a given criterion, such as the mean data serving time [1]. Knowing the time-variant popularity of the available items is vital to optimal scheduling. The process of deriving the popularity distribution and adjusting the schedule accordingly is called adaptation [2]**.** TV and radio broadcasting are typical application examples of adaptive scheduling. A broadcast station seeks to adjust the air times devoted to news, entertainment and advertisements in a way that satisfies the time-variant preferences of the clients. Other examples, such as information kiosks, are typically placed in crowded places (museums, airports) and fine-tune their schedules by monitoring the clients' preferences directly or indirectly [2].

Push-based broadcasting [3] relies solely on the knowledge of the data popularity distribution, disregarding individual client queries. Its opposite, pull-based process [4], [5], deals with individual, known client queries, serializing them in an optimal manner. The present study discusses adaptation issues in push-based broadcasting. In this case, the standard network topology consists of a central scheduling server and a number of wireless clients in its range. The server constructs a broadcast schedule comprising of several occurrences of each available data item. The schedule is then transmitted periodically over the covered area. During the broadcast of the schedule, the server collects lightweight, indirect feedback regarding the clients' satisfaction for the current schedule. This feedback can be exemplary derived from Facebook profile data (static) or aggregate client electromagnetic activity (dynamic) [2], [6], and is used for estimating the client preferences and adapting the schedule accordingly.

Static feedback typically requires manual processing and is not fitted for highly dynamic environments. On the other hand, the dynamic feedback approaches employ idealized physical layer information (aggregate electromagnetic activity near the server), assuming a linear relation to application layer attributes (i.e. data popularity). The present study proposes a novel adaptation scheme which requires seemingly incomplete information (i.e. monitoring few statistical moments of easily observable metrics) to reproduce the data item popularity distribution. To this end, the proposed study employs the principle of maximum uncertainty [7]. The approach ensures lightweight feedback requirements and privacy, as well as independence from physical-layer assumptions. Analysis and simulation results show improved efficiency, convergence speed and error tolerance over related approaches.

II. Related Work

Knowledge of the data item popularity has been a fundamental assumption of major studies on wireless push systems. Authors in [8] proved that the broadcast of each data item must occur periodically and proportionally to the square root of its popularity, in order to minimize the overall mean query serving time. This conclusion was analytically proven in [1], which also presented a lower bound for the overall mean serving time. The assumption of known item popularity is also the basis of recent studies on optimal data caching at the client devices [9], multichannel broadcasting [10], pull-push hybrid data dissemination [11], noise resilience and battery lifetime extension through the minimization of the clients' stream reading duration [12]. Furthermore, knowledge of the data popularity distribution is essential for minimizing the implementation cost of push systems [13].

The assumptions of wireless environment and highly dynamic client sets limit the use of e-voting/polling techniques to small, easily tractable cases (e.g. WLANs). MAC slot starvation and security issues of polling can require resolutions

Paper approved by M. Motani, the Editor for Network Information Theory and Cooperative Protocols of the IEEE Communications Society. Manuscript received September 14, 2011; revised May 14, 2012.

The authors are with the Department of Informatics, Aristotle University, Thessaloniki 54124, Greece (e-mail: {cliaskos, gp}@csd.auth.gr).

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

Digital Object Identifier 10.1109/TCOMM.2012.091712.110618

0090-6778/12$31.00 © 2012 IEEE

too complex for wide-area, highly-dynamic, wireless environments [14]. In addition, the advertising nature of broadcast-based applications may mean that the clients cannot express their preferences in advance through votes or queries. Authors in [2] overcame these issues by employing learning automata at the broadcast server. The wireless clients are required to emit an electromagnetic (E/M) pulse upon discovering a useful item in the broadcast stream. The automaton then maps the aggregate E/M activity to a data popularity distribution. Nevertheless, usual wireless propagation issues such as fading and interference, may easily cause the divergence of the automaton.

The novel adaptation scheme proposed in the context of the present work, offers more accurate estimation of the data popularity distribution, while having several advantages: a) The client feedback can be asynchronous. b) The feedback is received at the application layer and is not related to physical layer parameters. c) It provides analytically-optimal error resilience. d) Its convergence time is equal to that of an imaginary (yet unfeasible) polling/voting, thus being much smaller than that of a learning automaton. e) Is of lightweight nature, requiring only the logging of one or more statistical moments of the mean query serving time at the server side. Consequently, its complexity is also tunable: logging more moments adds precision and complexity to the overall process. f) It can be combined with existing, learning automata-based techniques in a cross-layer fashion for improved performance.

The present work extends [15], in which entropy-based techniques were efficiently used for adaptation purposes. The original study considered the monitoring of just a single statistical moment and assumed evenly-sized data items and noiseless environments. These limitations are alleviated in the present study.

III. Prerequisites

A. Notation and Standard Assumptions

We regard a set of N data items arbitrarily indexed by $i = 1...N$. Each item i is associated with its size l_i (in $bytes$) and its popularity p_i, $\sum_{i=1}^{N} p_i = 1$. It is clarified that the item popularity metric, p_i, represents the percentage of the total queries that refer to item i over an infinite time horizon. Finally, $u_i \in \mathbb{N}^*$ will denote the number of periodic occurrences of item i in the employed broadcast schedule. In other words, the quantity L/u_i denotes the broadcast period of item i, L being the total size of the schedule.

No assumptions are made concerning the nature of a data item during the analysis. In accordance with the related work, an item is simply a piece of information that a client may acquire through a single query. It is clarified that in push-based, periodic broadcasting, the term "*client query*" does not imply posting a request to a server, but rather waiting for the broadcast of a specific item. According to [1], [2], [8]–[13], [15]–[18]: a) The assumption of wireless environment is used for justifying synchronous data stream reading by all clients, as well as the prohibition of polling techniques. Wireless data broadcasting typically serves hundreds of clients per coverage, and explicit client polling can easily degrade the operation of the system at MAC layer. b) Wireless coverage issues are idealized in order to focus on the evaluation of the adaptation process only.

According to [1], the waiting time for a query is defined as the interval from the moment it is expressed to the moment when the broadcast of the corresponding item begins**.** Assuming an infinite amount of queries and Poisson arrivals, the expected mean serving time is equal to**:**

$$\overline{W} = \frac{1}{2} \cdot \left(\sum\nolimits_{i=1}^{N} u_i \cdot l_i \right) \cdot \left(\sum\nolimits_{i=1}^{N} \frac{p_i}{u_i} \right) \tag{1}$$

Notice that $\overline{W}$ does not depend on the number of clients, which are handled collectively as a Gaussian process via the central limit theorem [1]. In addition, (1) measures $\overline{W}$ in size units (e.g. bytes). Conversion to time units requires knowledge of the physical transmission rate. The lowest possible value of the mean serving time ($\overline{W}$) is:

$$\overline{W}_{min} = \frac{\left(\sum_{i=1}^{N} \sqrt{p_i \cdot l_i} \right)^2}{2} \tag{2}$$

which is produced when ([1]):

$$u_i \propto \sqrt{\frac{p_i}{l_i}}, i = 1 \ldots N, \tag{3}$$

The u_i quantities are collectively referred to as *item occurrence ratio* [1]. The bound of equation (2) is achievable in practice, as shown in [1], [13]. Equation (3) is known as *the square root rule*. Finally, the total size of the schedule is expressed as:

$$L = \sum\nolimits_{i=1}^{N} u_i \cdot l_i \tag{4}$$

which is minimized when $u_i = 1, \forall i$.

B. The principle of Maximum Entropy (Uncertainty).

The principle of Maximum Entropy refers to the procedure of defining a discrete probability distribution that maximizes the Shannon Entropy under a set of constrains [19, p. 573]. It is used for extracting the most probable probability distribution under the absence of sufficient knowledge for precise definition. (E.g. having a system of N variables and $M < N$ equations).

Assume a discrete random variable X and its value set $\{x_1, \ldots, x_N\}$, with probabilities $\{p_1, \ldots, p_N\}$. Let there be M constrains, c_j, $j = 1 \ldots M$, of the form $f_j(p_1, \ldots, p_N, x_1, \ldots, x_N) = 0$. The objective is to find the probability distribution p_i, $i = 1 \ldots N$ that meets all constraints c_j, $j = 1 \ldots M$, while maximizing the Shannon entropy :

$$H = -\sum_{i=1}^{N} p_i \cdot ln\left[p_i\right] \tag{5}$$

Should the constrains c_j refer to keeping the statistical moments of X equal to some observed values, it will hold that:

$$c_j :< x^j > - \sum_{i=1}^{N} x_i^j \cdot p_i = 0, \; j = 1 \ldots M \tag{6}$$

where $< x^j >$ denotes the observed j^{th} moment. (Note that $< x^1 >= \overline{W}$, i.e. the mean value). The problem is

traditionally solved by using Lagrange multipliers, and the detailed procedure can be found in [19, p. 579] (among other sources). Finally, it is derived that:

$$p_i = \frac{\prod_{j=1}^{M} A_j^{x_i^j}}{\sum_{i=1}^{N} \prod_{j=1}^{M} A_j^{x_i^j}} \tag{7}$$

where A_j corresponds to the Lagrange multiplier*[1] of constraint c_j. The proper A_j values are numerically calculated in order to uphold the existing constraints, c_j, $j = 1 \dots M$.

IV. The Proposed Entropy-based Adaptation Scheme

Mapping the broadcast scheduling problem to the maximum entropy formulas is straightforward. Equation (1), expressing the mean query serving time, can be rewritten as:

$$\overline{W} = \sum_{i=1}^{N} \left(\frac{L}{2 \cdot u_i}\right) \cdot p_i \tag{8}$$

By setting $x_i = (\frac{L}{2 \cdot u_i})$, $i = 1 \dots N$, equation (8) directly corresponds to constrain c_1 (known mean value) of equation (6). Thus, by monitoring the real mean query serving time and employing the procedure of Section III-B, an estimation of the data item popularity, p_i, can be obtained via equation (7).

In order to monitor the mean serving time, each client sends his personal waiting time back to the server as feedback. The server calculates the mean value of all received replies, obtaining the needed approximation of the statistical moment.

However, this direct approach faces some practical difficulties:

- Broadcasting applications usually involve data advertising. Thus, a client may not know in advance that an item is of personal interest, prohibiting the logging of his waiting.
- Producing an accurate estimation of the statistical moments may require large amounts of feedback.

The aforementioned difficulties can be overcome by employing the *Virtual Broadcast Schedules*, which are introduced in the context of the present work.

A. Virtual Broadcast Schedules, Real Client Feedback and Moment Monitoring

Virtual broadcast schedules are used only for monitoring statistical moments of the query serving time. The monitoring is not intended to be *accurate*, but *uniquely representative*. Furthermore, virtual schedules need not be actually constructed through a serialization process (e.g. [13]), since they do not serve a purpose in real data transmission.

A virtual schedule comprises periodic occurrences of each contained data item: the interval between consecutive occurrences of each item in the schedule is constant. Spacing conflicts between items are irrelevant, due to the virtual nature of the schedule. Despite its physical size, any data item is considered to have a constant, unary size inside the virtual schedule. Finally, assuming that the real data popularity distribution is unknown, the number of occurrences, u_i, of each item is set to a unique value. (In practice, u_i is defined by assuming a nearly flat distribution, $p_i \approx 1/N$, $i = 1 \dots N$, where $p_i \neq p_k$, $\forall i \neq k$, and applying eq. (3)).

In periodic schedules, the mean serving time for an item i is equal to half the interval between two consecutive occurrences ([1]):

$$\overline{w_i} = \frac{1}{2} \cdot \frac{L}{u_i} \tag{9}$$

where L is the total size of the virtual schedule (equation (4), for $l_i = 1, \forall i$). This fact can be directly exploited to overcome the previously mentioned difficulties:

Consider an actual data broadcast occurring by a real schedule ([1], [13]), which contains the same u_i item occurrences as the virtual one. Having finished processing a useful item, each client sends the item index, i, back to the server as feedback. The server then simply aggregates the corresponding, virtual mean serving times (equation (9)) and produces the required statistical moments. The uniqueness of item occurrences, u_i, $\forall i$, ensures that all items contribute in a discrete, non-overlapping manner.

The proposed approach of virtual scheduling and index-based feedback alleviates the drawbacks of the direct, waiting time aggregation solution:

- The client does not log his waiting time. The proposed process requires feedback *a posteriori,* enabling its use in advertising applications.
- The client responses can be asynchronous and independent of physical layer attributes, contrary to the learning automata techniques of [2], [18].

Claim 1. Virtual scheduling requires less client feedback for convergence than the direct approach.

Proof: Assume a number of R received client responses, $\{r_1, \dots r_k, \dots r_R\}$. According to the direct approach, the responses will represent query serving times:

$$r_k \in [0, \max\{L/u_i, i = 1 \dots N\}] \tag{10}$$

On the other hand, assuming virtual scheduling, the responses will represent *mean* query serving times (equation (9)):

$$r_k^* \in \{L/2 \cdot u_i | \, i = 1 \dots N\} \tag{11}$$

Therefore, it will hold that:

$$r_k = r_k^* + \epsilon_k \tag{12}$$

which expresses the simple fact that the serving time for a query fluctuates around its mean value, with the error ϵ_k denoting the fluctuation. Summing over all k and dividing by R produces:

$$\frac{\sum_{k=1}^{R} r_k}{R} = \frac{\sum_{k=1}^{R} r_k^*}{R} + \frac{\sum_{k=1}^{R} \epsilon_k}{R} \tag{13}$$

The first term represents the mean serving time extracted by the direct approach ($\overline{W_d}$), the second by the virtual scheduling

[1]*According to [19, p. 579], $A_j = e^{\lambda_j} > 0$, λ_j being the Lagrange multiplier. However, since there exists a $1-1$ correspondence, A_j will be referred to as Lagrange multiplier for the sake of simplicity.

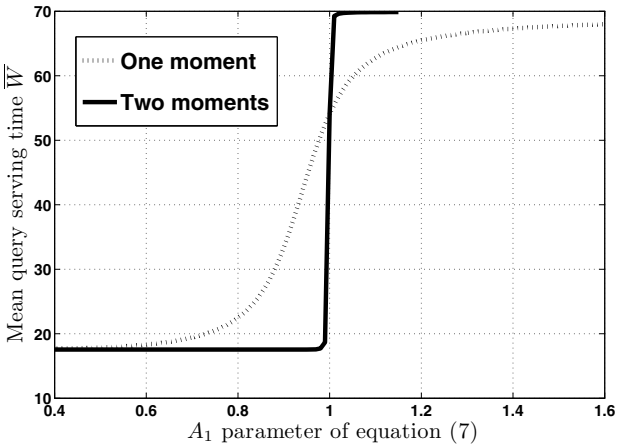


Fig. 1. [Numerical example] Monitoring one and two moments and effects on the error tolerance of the proposed adaptation scheme. In the case of single-moment monitoring, large variations of the monitored moment (mean query serving time, y-axis) correspond to large variations of the A_1 parameter. Therefore, the estimated data popularity distribution p_i is greatly affected as well, through equation (7). On the other hand, two moments offer increased error tolerance: variations of the monitored moment have a trivial impact on the A_1 parameter.

($\overline{W_v}$) and the third the mean error ($\overline{\epsilon}$). With $\overline{W}$ denoting the real mean serving time, it is produced:

$$\overline{W_d} - \overline{W} = \overline{W_v} - \overline{W} + \overline{\epsilon} \tag{14}$$

Equation (14) states that, given R responses, the direct approach expands the error interval of the virtual scheduling by an additional error factor $\overline{\epsilon}$. Therefore, the time required to converge with the same precision is lower for the proposed approach. This concludes the proof. ■

Finally, it is noted that the assumption of unary item sizes in virtual scheduling may alter the monitored value of the statistical moments (e.g. equation (1)) by a constant factor. However, the output of the adaptation process (i.e. the data item popularity) does generally not depend on item sizes or how these are measured. Therefore, the following proposition is formulated:

Proposition 2. *Altering the item sizes in a virtual schedule does not have an impact on the precision of the adaptation, provided that the data item occurrences, u_i, remain unique $\forall i$.*

As a simple example, consider the case of adding redundant, null data at the end of each data item, making them equal in size. The process would obviously not make an item more or less popular. Therefore, unary item sizes in the virtual schedule are retained for simplicity.

The unique initial values of u_i favor the 1-1 correspondence between the monitored statistical moments and the produced popularity distribution. Conceptually, the issue is similar to finding the solutions to an equation of the form $u_1 \cdot x_1 + u_2 \cdot x_2 = C$ for given C, u_1, u_2. When $u_1 = u_2$ the system will have no means of differentiating between x_1 and x_2.

B. Effects of multi-moment monitoring on error resilience

Monitoring more than one statistical moment can imbue the proposed scheme with increased resilience to feedback errors. As expressed by equation (7), the estimated popularity, p_i, of a data item i is defined by the Lagrange parameters A_j, $j = 1 \dots M$. Thus, low sensitivity to feedback errors means that large variations of monitored moments (e.g. the mean query serving time) have a small impact on the resulting A_j, $j = 1 \dots M$ parameters. This concept is illustrated in the numerical example of Fig. 1. The configuration comprises $N = 5$ data items, following a Zipf [20] popularity distribution (p_i) with skewness $\theta = 0.3$. We assume that the monitoring of the mean query serving time has been accomplished, and its value set is represented by the x-axis. In other words, we provide as input hypothetically monitored values of the mean query serving time. For each possible value on the x-axis, the principle of Maximum Entropy (Section III-B) is applied**.** The produced A_1 parameter values are then logged and plotted versus the corresponding mean serving time. When monitoring a single moment, the resulting relation between $\overline{W}$ and A_1 shows that fluctuations of the mean serving time cause similar changes to the A_1 parameter as well. However, when the process is repeated for the two-moment monitoring case, changes in the monitored mean serving time have little to no effect on A_1 and, consequently, to the p_i distribution as well. This indication of error tolerance is analytically proven and formalized in the following theorem:

Theorem 3. *When more than one statistical moments of the query serving are monitored, their required precision for EAS convergence decreases.*

Proof: See Appendix. ■

The theorem states that the effects of noise in the monitored values can be limited by monitoring more statistical moments. In push-based systems, client feedback is not critical to the query serving process. Therefore, a good amount of it can be lost due to absence of error control (e.g. using UDP to send feedback to the server) or user negligence. These factors limit the accuracy of the monitored statistical moments and hinder the convergence of EAS. Monitoring more statistical moments enables EAS to operate under presence of higher error levels.

C. Formulation of the proposed scheme

The conclusions of the preceding analysis are gathered, and the *Entropy-based Adaptation Scheme (EAS)* is formulated in algorithmic format (Algorithm 1). The algorithm requires a sorting of the data items by descending expected popularity. Since a statistical moment bears no indexing information. EAS initializes the p_i as a nearly flat p.d.f. (e.g. a Zipfian p.d.f. with a skewness factor of 0.01). The corresponding real and virtual schedules are constructed next. Finally, EAS proceeds to gather client feedback (returned item indexes) and calculate the statistical moments of the query serving time. The final estimation of the popularity distribution is produced by solving equations (6) and (7) numerically. The present study uses the classic Levenberg-Marquardt [21] algorithm.

The choice of the numerical solution method also determines the complexity of the EAS adaptation procedure. The computational requirements of EAS stem from the need to solve numerically the equations (6) and (7), i.e. finding a solution to a $(N - M) \times (N - M)$ non-linear system

where N is the number of items and M the number of monitored moments. The problem is reduced to minimizing a $N \times M$ solution gradient. According to [22], a lax upper bound of required iterations is $O(\epsilon^{-2})$, ϵ being the acceptable precision of the solution. Therefore, precision of the scale of $10^{-4} - 10^{-5}$ can be achieved in trivial time ($< 1sec$), while being adequate for the EAS process. Moreover, more recent versions of the algorithm are known to achieve better precision in smaller computational times [22].

Algorithm 1 Entropy-based Adaptation Scheme (EAS)

INPUT: The number of logged moments, M; An indexing, I, of the items by descending popularity.

OUTPUT: The item probability distribution estimate: $p_i, i \in I$.

```
1  Initialize array FEEDBACK=[].
2  Initialize (1xM) array MOMENTS=[0,0,..,0].
3  Initialize variable feedbacks_rcvd=0.
4  Initialize array p[i] as near-flat p.d.f..
5  Construct corresponding real schedule, RBS, e.g.[10]
6  Construct corresponding virtual schedule, VBS.
7  Initialize array w[i] through equation (9).
8  FORK: Broadcast RBS, FEEDBACK.push(recvd. feedback)
9  REPEAT
10         FOR new entry, k, in FEEDBACK
11                feedbacks_rcvd++.
12                MOMENTS[j]+=w[I[k]]^j, j=1..M.
13                FEEDBACK.pop().
14         ENDFOR
15 UNTIL (all items have been broadcast at least once).
16 MOMENTS[j]/=feedbacks_rcvd, j=1..M.
17 Update p[i] by solving equations (6),(7) numerically.
```

V. Simulations

In this section, the proposed scheme is compared to related push and pull-based broadcast techniques. Regarding the push-based counterparts, the Classic Learning Automata (C-LA) adaptation scheme of [6] constitues the base for all related adaptive schemes in wireless push systems. The operation of C-LA relies on the physical layer-aware client feedback which was described in Section II. The Interpolated Learning Automata (I-LA) of [18] epitomize convergence time for this class of adaptation schemes. The scheme assumes the item popularity ordering to be a given, just like the proposed EAS algorithm. The adaptation process begins by periodically broadcasting only a small portion of the total items, attempting to extract their exact probabilities. Cubic spline interpolation is used for approximating the popularity of the remaining items. The comparison tests the efficiency, convergence time and error resilience of the compared approaches**.** Once the comparison among push-based schemes has been accomplished, the prevailing technique is compared to pull-based broadcasting (on demand). On demand broadcasting requires the clients to send explicit queries to the server. The server then schedules the data broadcast according to the current queue of queries. With exact knowledge of the client queries, the server has no further need for adaptation schemes. The comparison considers classic pull-based schedulers (First Come First Serve-FCFS, Longest Wait First-LWF [23], RxW and variations [24], Summary Tables On-demand Broadcast Scheduler-STOBS [25]) and a more recent approach (Slack time Inverse Number of pending requests-SIN-a [4]). An overview of pull-based approaches is given in [26]. The comparison with pull-based solutions is oriented towards performance (serving time) versus scalability.

A. Comparison of Push-based solutions

The comparison of push-based solutions follows the standard assumptions of the field. We regard a star topology of a broadcast server and 300 wireless clients. Coverage issues are idealized in order to focus on the efficiency of the adaptation process only. All clients listen to the broadcast stream synchronously.

The server maintains a database of N items whose popularity follows a predefined probability distribution function (p.d.f). The Zipf p.d.f. is the de facto standard in the field and has been studied extensively [1], [8]–[13], [16], [17]. However, additional distributions are examined as well. Item sizes are randomly picked from a predefined range.

The channel broadcast rate is traditionally set to one unit, while the item sizes are considered unitless and normalized in $(0, 1]$. This standard assumption has been introduced by the related studies for generality purposes, since no physical context is strictly adopted. The reader may multiply the produced waiting times by an appropriate $^{data\,size\,unit}/_{channel\,rate}$ constant in order to translate them to physical time units of interest.

The simulations are initialized as follows. A number of $300,000$ client queries are produced according to the chosen, real popularity distribution and are distributed equally to the clients. A client may have one query pending at any given moment, and a random *Think Time* of $[0, L/N]$ between successive queries. Initially, the server has no knowledge of the real distribution and assumes a near flat one in any case (Zipfian with $\theta = 0.01$). The compared adaptation processes gather feedback and re-estimate the probability distribution as described. The simulation ends with the depletion of all queries.

Specific values of the aforementioned simulation parameters are given per figure. All runs are repeated 100 times in order to extract dependable mean values.

Finally, the real scheduler of [13] has been shown to surpass the alternatives [3], [27], and is used where required during the proposed EAS procedure. Furthermore, two-moment monitoring produced optimal results in all test cases, rendering the use of higher order moments redundant. The simulator was implemented in MATLAB^{TM}. EAS uses the MATLAB^{TM} implementation of the Levenberg-Marquardt method.

1) Efficiency potential: A set of $N = 10$ items and 300 clients are assumed. The number is purposefully small for illustrative reasons, but the produced conclusions hold for all test cases. The item sizes are random, $l_i \in (0, 1]$. Their popularity distribution is Zipfian with $\theta = 0.7$. The proposed EAS (single-moment), and the counterparts C-LA, I-LA are set to run in a noiseless environment. The goal is to study the convergence precision of each scheme. The finally achieved mean query serving time is logged as a metric of convergence accuracy.

A representative example of all tested random p.d.f. is shown in Fig. 2. In an effort to minimize the mean query

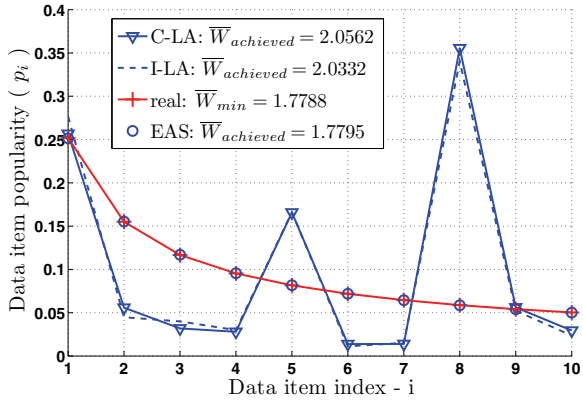


Fig. 2. Accuracy potential of the proposed EAS (single-moment) and the C-LA and I-LA adaptation schemes, with regard to the real p.d.f. in question. Items have been sorted by real popularity for clarity. The proposed EAS converges to the real p.d.f. with near-perfect accuracy. The learning automata techniques exhibit considerable divergence from reality. The items have random sizes: [0.4364, 0.8403, 0.6785, 0.6790, 0.0634, 0.7659, 0.4045, 0.0116, 0.0811, 0.1890].

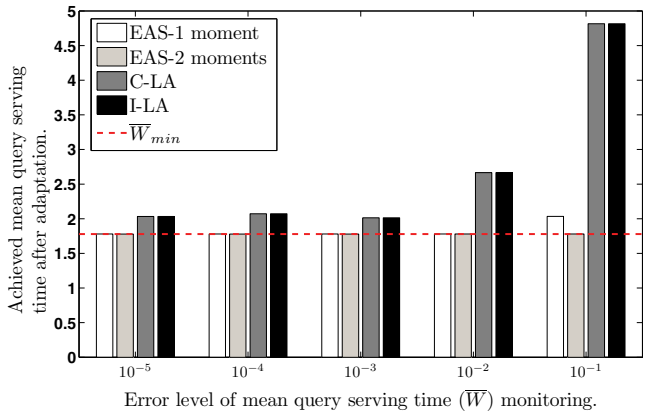


Fig. 3. Error resilience of EAS, I-LA and C-LA. EAS recovers successfully from higher levels of environmental noise, i.e. erroneous or lost feedback attributed to wireless propagation or security issues (e.g. fraudulent client voting). The convergence is closer to the theoretical minimum as well.

serving time, both C-LA and I-LA get quickly locked into a suboptimal local minimum. Both schemes converge near the theoretical lower bound ($\overline{W}_{min} = 1.7788$) of equation (2), but fail to do so with adequate precision. EAS, on the other hand, converges with high accuracy, as indicated by the coinciding plots and mean times of EAS and REAL.

2) Error Tolerance: The previous experiment is repeated, assuming presence of errors. A portion of the environmental feedback is lost, causing tunable, $\sim [10^{-5}, 10^{-1}]$ precision errors in the monitored mean serving time. Two versions of EAS (single-moment and two-moment versions) compete with C-LA and I-LA in terms of converging to the lower bound of equation (2). The results are shown in Fig. 3.

Both I-LA and C-LA can withstand errors up to the scale of 10^{-3}. EAS (single-moment) achieves near-optimal accuracy for larger errors, up to 10^{-2}. EAS (two-moments) converges for errors even at 10^{-1} level, as dictated by Theorem 3 for improved error resilience when monitoring more moments. Notice that the real value of $\overline{W}_{min}$ is just one scale

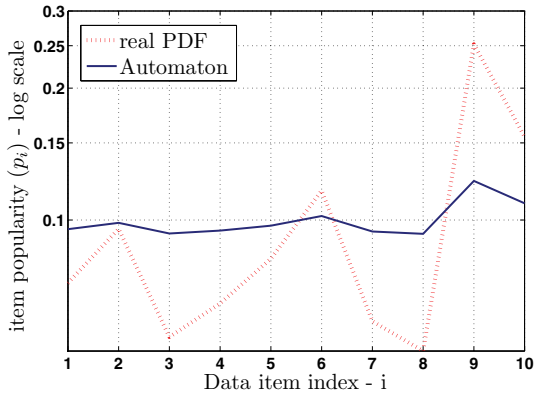


Fig. 4. After one broadcast of a forced-flat broadcast schedule, C-LA retrieves the correct data item indexing which can be used an an input for I-LA and EAS. This fact encourages the cross-layer collaboration of the proposed EAS scheme and C-LA: C-LA recovers the correct data item indexing by processing physical layer information, while EAS handles the estimation of the popularity distribution at the application layer.

(10^0 : 1.7788) greater than the tolerated error level of EAS (10^{-1}). Finally, all schemes stop converging for errors higher than 10^{-1} (i.e. when the noise completely overwhelms the monitoring). This holds for any number of moments.

The performance of EAS and I-LA also depends on the initial item indexing (sorting by popularity) [15]. However, the risk of erroneous item indexing can be reduced to insignificant levels for both EAS and I-LA, even when nothing is known of the right sorting. Consider Fig.4, where the C-LA automaton attempts to converge in a single broadcast cycle containing only one instance of each item (flat broadcast). While convergence is by no means achieved, the indexing is successfully retrieved with perfect accuracy. The time required is equal to that of an imaginary polling, i.e. a single broadcast of each item. Thus, C-LA can be integrated to I-LA and EAS in a cross layer-logic: C-LA simply hands over convergence to I-LA or EAS after a forced-flat broadcast. An additional advantage of this approach is that the automaton can process physical layer-related feedback (i.e. the aggregate received power of client replies [2]), while EAS can work in parallel at application layer, monitoring the required statistical moments. Optimizing this cross-layer collaboration is a subject of future work.

3) Convergence time: In order to check the convergence time of the compared schemes, it is assumed that the item popularity distribution is unknown. For $N = 100$ items with random sizes ($l_i \in (0, 1]$)) and a Zipfian real p.d.f. ($\theta = 0.7$), all schemes attempt to converge to the lower bound of the mean serving time (equation (2)). Their performance is logged at every client response, which is received at each individual item broadcast. The metric of the performance is the mean serving time that would have been achieved, should the convergence had stopped at the specified time moment (equation (1) for u_i obeying to equation (3) where p_i is the current popularity estimate). The results are shown in Fig. 5.

C-LA converges to a suboptimal local minimum after approximately 500 item broadcasts in a noiseless environment. Its convergence is straightforward, with no intermediate steps. I-LA and ESA start with a flat, one-time broadcast of all 100

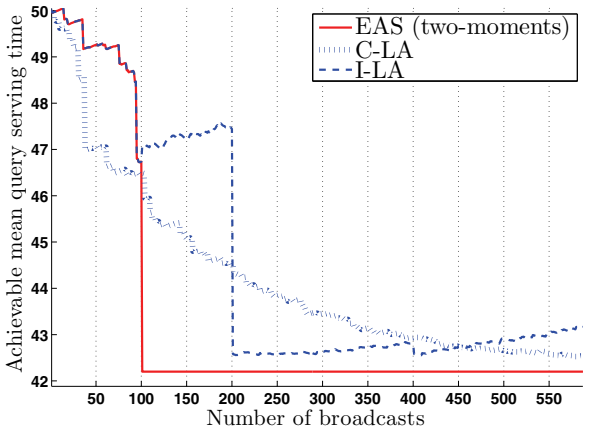


Fig. 5. Converge as a function of time (number of item broadcasts). The proposed EAS converges faster and closer to the theoretical minimum of equation (2) (start of y-axis). Test configuration is $N = 100$ items, with Zipfian p.d.f. with a skewness of $\theta = 0.7$. Similar results are extracted for more or less skewed distributions.

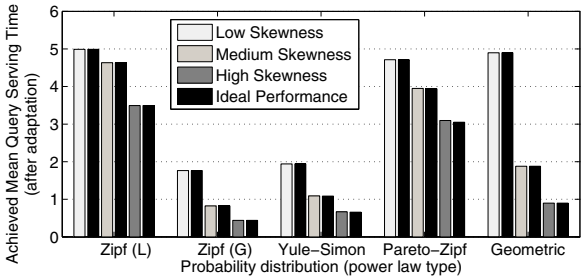


Fig. 6. The behavior of the proposed adaptation scheme as a function of the probability distribution that governs the popularity of the data items. Three skewness variations are examined per distribution. The corresponding parameters for each case (low, medium, high skewness) are given in Table I. After the convergence of the adaptation scheme, the push-based data dissemination scheme achieves near-optimal performance (black bars) in all tested cases.

TABLE I
PROBABILITY DISTRIBUTION FUNCTIONS AND PARAMETERIZATIONS CORRESPONDING TO THE RESULTS OF FIG. 6.

Distribution	Skewness parameters		
	Low	Medium	High
Zipf (L)[1]	$\theta = 0.1$	$\theta = 0.5$	$\theta = 0.95$
Zipf (G)[1]	$\theta = 1.5$	$\theta = 2$	$\theta = 2.5$
Yule-Simon[2]	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$
Pareto-Zipf[3]	$\theta = 0.5$, $n = 1$	$\theta = 0.95$, $n = 2$	$\theta = 1.5$, $n = 3$
Geometric[4]	$\beta = 0.01$	$\beta = 0.1$	$\beta = 0.2$

[1] $p_i = \frac{i^{-\theta}}{\sum i^{-\theta}}$, [2] $\rho \cdot Beta(i, \rho + 1)$, [3] $\frac{(i+n)^{-\theta}}{\sum (i+n)^{-\theta}}$, [4] $\beta(1-\beta)^{i-1}$

items in order derive the initial indexing of the items, since it is a required input for both schemes. Since the program is fixed to being flat, their convergence is slower than that of C-LA at this stage. According to [18], I-LA proceeds to broadcast a 10% of the items in order to get an accurate approximation of their popularity. Notice that the remaining 90% of the items are not broadcast at all at this stage. Their popularity is estimated by performing interpolation on the advertised 10% of the items. This means that the overall mean serving time increases indefinitely due to clients waiting for these non-broadcast items. The increase rate is proportional to the total number of active clients. Furthermore, the broadcast of the 10% of the items must be repeated for several cycles (here 10 times) in order to extract reliable popularity values. It is not, however, guaranteed that the approximation of the popularity of the 10% will be accurate (see for example Fig. 2). Nevertheless, we assume the best-effort case, where the approximation is precise. The interpolation intervenes and the convergence accelerates, approaching a local minimum. However, the automaton is prone to seeking other local minima nearby and starts to diverge. On the other hand, the proposed EAS converges sharply, nearer the exact theoretical minimum, once the indexing has become available.

4) Effects of different popularity distributions: Regarding the modeling of data popularity, the Zipf distribution has been used almost exclusively in the related studies on push-based broadcasting [1], [2], [8]–[12], [15]–[18]. The Zipf distribution has been observed to describe adequately realistic popularity distributions [28]. However, studies indicate that other power laws may be more fitting for specialized contexts [29]. We therefore proceed to study the effects of different distributions on the efficiency of EAS.

Returning to the standard simulation setup (section V-A), we assume $N = 100$ items with unary sizes. We study each of the popularity distributions present in Table I. Three skeweness parameterizations are considered per case. The set of distributions also includes two cases of the Zipf distribution corresponding to moderately skewed (ZIPF(L)-θ Less than 1) and highly skewed (ZIPF(G)-θ greater than 1) distributions.

The simulations consider no initial knowledge of the current distribution, but known item ranking. Once EAS has converged, we begin to measure the mean query serving time until the depletion of all queries. The achieved mean value is compared to the theoretical expectation of equation (2). The results are given in Fig. 6.

EAS converges successfully, achieving performance practically equal to the theoretical optimum. The results provide good evidence towards the affinity of EAS for power law-based distributions. Notice that EAS is not expected to operate equally well for other types of distributions (e.g. linear). In such cases, EAS will essentially attempt to find a best-matching power law. However, power laws are known to describe realistic client behavior well, especially in the large scales considered by push-based broadcasting.

B. Comparison with pull-based approaches

Returning to the simulation topology, we consider a server with a downlink ($0.5MBps$) and an uplink ($0.5MBps$) channel. the downlink is divided to 100 MAC slots that can operate concurrently. When sending a query, a client scans instantaneously all slots, looking for a free one. If no slot is found, the client backs off for a random time $[0, 5]sec$ and retries. Upon successful connection, a query

packet ($100Bytes$) is forwarded to a queue of queries awaiting ACK with a maximum size of 100 queries. The ACK queue empties with a rate of 1000 queries per second. If the queue is full, the query packet is lost and the client experiences a denial of service event. After a timeout interval of $10sec$ with no ACK from the server, the client attempts to reconnect and send his query. Queries that have successfully entered the ACK queue are also registered to a SERVICE queue, awaiting to be processed on the next item broadcast decision. The clients constantly monitor the broadcast stream for needed items.

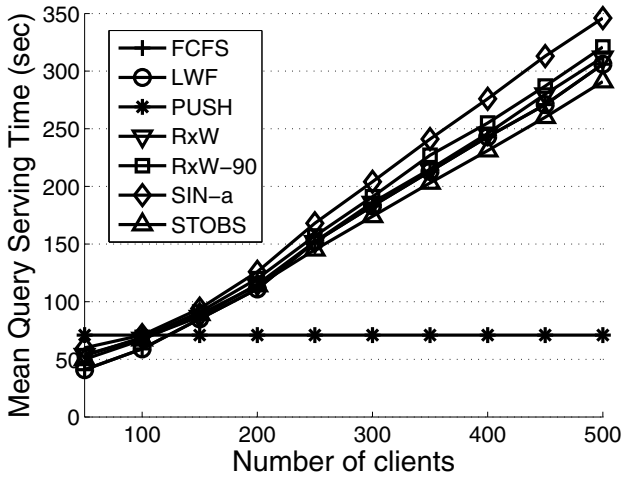
We assume $N = 100$ data items items with uniform sizes ($1MB$). Their popularity distribution is Zipfian with $\theta = 0.5$. Each client is assigned 500 queries with a maximum interarrival time ($ThinkTime$) of $10sec$. The total number of users is an input parameter. The achieved mean query serving times of the pull and push-based methods are presented in Fig. 7. Push-based serving times are logged after the adaptation process (EAS).

As shown in Fig. 7a, pull-based schemes are superior under absence of congestion. However, when the number of users exceeds the available MAC slots, MAC back-off and denial of service (DoS) events take place. While MAC slot starvation events are steadily increasing with the number of users (Fig. 7b), DoS events reach a certain limit and remain approximately constant thereon (Fig. 7c). At that point the MAC slot usage has reached its maximum, leading to a steady rate of incoming ACK requests and, consequently, DoS events as well.

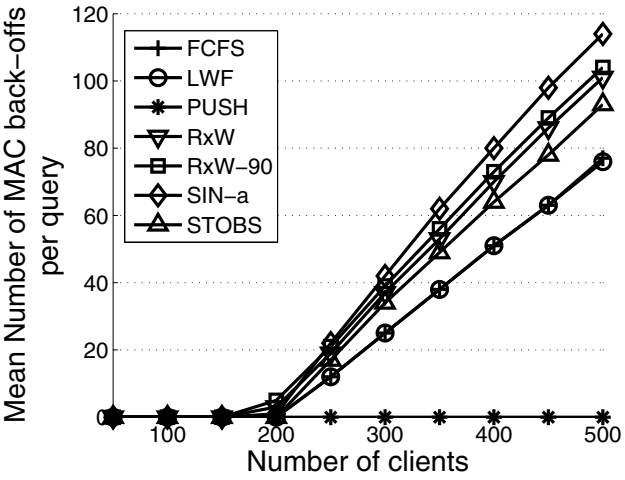
The results reinforce the conclusions of previous studies [13]. Push-based broadcasting is a cost-effective solution for scalable data dissemination. There will always exist a powerful (and expensive) enough, pull-based counterpart that can provide superior performance. In the simulation scenario for example, a well built content distribution network (CDN) would overcome MAC-layer issues. Choosing between a push and a pull-based solution is a matter of context-variant balance between performance and expenses.

VI. Conclusion

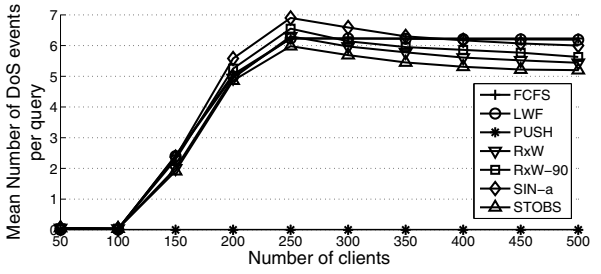
In the context of this paper, a novel adaptation scheme for extracting client preferences in a wireless broadcast environment was presented. The proposed, Entropy-based Adaptation Scheme (EAS) is a lightweight procedure that can reproduce a complete data popularity distribution just from monitoring the mean waiting time (or more statistical moments) over all clients. Thus, EAS can ensure client privacy in addition to improved efficiency, convergence speed and error tolerance over existing state-of-the-art approaches. Future steps are directed towards combining EAS with physical layer-aware learning automata, in order to produce more efficient, cross-layer adaptation techniques.



(a) The mean query service time achieved by push and pull-based dissemination methods, as the number of users scales up. Push-based broadcasting provides scalability regardless of the number of users. On the other hand, the performance of the pull-based schemes is impaired significantly as the number of clients exceeds the maximum amount of MAC slots (100).



(b) When a client finds no available MAC slot for connecting to the server, he backs off for a random time interval and retries. This interval is added to the query service time. As the number of users escalates, a user may have to perform several tenths of MAC connection attempts per single query. Push-based dissemination is an exception, since it operates without explicit client queries (and, consequently, without client-server connections either).



(c) Once a client has established a MAC-level connection to the server, his query may not be processed due to the service queue being full (denial of service-DoS). Without ACK from the server, the connection times out and the client attempts to re-connect and send his query. As the number of users grows, a client may experience 5-7 DoS events prior to a successfully received query. After a certain point (≈ 250 clients) MAC slot starvation becomes the main bottleneck, and the mean amount of DoS events remains constant.

Fig. 7. Comparison of scalability and performance between the push-based and pull-based solutions.

APPENDIX

Proof of Theorem 3.

Assume that a monitoring period has just ended, and the statistical moments $< x^j > \; j = 1, 2, 3, ...M$ have been logged. Consider two feedback processing servers. The first was monitoring just $M = M_1$ statistical moments, while the second one logged $M = M_2 > M_1$ moments during the adaptation process. The common moments, $< x^j >$, $j = 1...M_1$, are equal in both cases. As explained in the context of Fig. 1, high error tolerance means that large variations of the observed $< x^j >$ moment cause little effect on the A_j parameters of equation (7). Therefore, the claim of increased error tolerance with the addition of more monitored moments is expressed as follows:

$$\left|\frac{\partial < x^j >}{\partial A_k(M_2)}\right| > \left|\frac{\partial < x^j >}{\partial A_k(M_1)}\right|, \; k = 1 \dots M_1, \; j = 1 \dots M_1 \tag{15}$$

The notation $A_k(M)$ expresses the k^{th} of the A_j parameters of equation (7), in the case of M-moment monitoring. We employ the utility function (21) to write:

$$\begin{aligned}\left|\frac{\frac{\partial < x^j >}{\partial A_k(M_2)}}{\frac{\partial < x^j >}{\partial A_k(M_1)}}\right| &\overset{(21)}{=} \left|\frac{\frac{1}{A_k(M_2)}\left(< x^{j+k} > - < x^j >< x^k >\right)}{\frac{1}{A_k(M_1)}\left(< x^{j+k} > - < x^j >< x^k >\right)}\right| \\ &= \left|\frac{A_k(M_1)}{A_k(M_2)}\right| \end{aligned} \tag{16}$$

Consequently, in order to prove Theorem 3, one needs to prove the following equivalent inequality:

$$\left|\frac{\frac{\partial < x >}{\partial A_1(M)}}{\frac{\partial < x >}{\partial A_1(1)}}\right| = \left|\frac{A_1(1)}{A_1(M)}\right| > 1 \tag{17}$$

i.e. the application of inequality (15) and equation (16) for the cases of $M_1 = 1$ moment and $M > 1$ moments. Notice that only the first moment (mean value, $< x >= \overline{W}$) is common in both cases. Therefore, the j, k parameters of (16) have been set to 1. The claim of (17) is proven as follows.

Given that the adaptation process has correctly converged to approximately the same popularity distribution (p_i) in both cases (single and multi-moment monitoring), it will hold through equation (7) that:

$$p_i : \frac{\prod_{j=1}^{M} A_j^{x_i^j}(M)}{\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)} \approx \frac{A_1^{x_i}(1)}{\sum_{i=1}^{N} A_1^{x_i}(1)}, \; i = 1 \dots N \tag{18}$$

Since equation (18) must hold for ever $i = 1 \dots N$, it will hold for any data item $k \neq i$ that:

$$\begin{aligned}\frac{p_i}{p_k} : A_1(M)^{x_i - x_k} \cdot \prod_{j=2}^{n} A_j(M)^{x_i^j - x_k^j} &\approx A_1(1)^{x_i - x_k} \\ \Rightarrow \frac{A_1(1)}{A_1(M)} &\approx \prod_{j=2}^{n} A_j(M)^{\frac{x_i^j - x_k^j}{x_i - x_k}}\end{aligned} \tag{19}$$

Equation (19) must also hold for any x_i, x_k, $i \neq k$ pair and, consequently, $A_j(M) \to 1$, $\forall j$. (The reader may validate this claim by calculating the first derivative ($\partial(\cdot)/\partial x_i$) of both parts of equation (19) and solving the resulting equation). Therefore, with no loss of generality, there also, always exist complete solutions of the form $A_j(M) \to 1^+$, $\forall j$. Furthermore, it is valid to assume that $x_i >> 1$, $\forall i$, since it represents the interval between consecutive broadcasts of item i, measured in bytes. Therefore, from (19) it is derived that:

$$\left|\frac{A_1(1)}{A_1(M)}\right| > 1 \Longleftrightarrow \left|\frac{\partial < x >}{\partial A_1(M)}\right| > \left|\frac{\partial < x >}{\partial A_1(1)}\right|, \quad (QED) \tag{20}$$

Utility functions

We will prove that:

$$\frac{\partial < x^j >}{\partial A_k(M)} = \frac{1}{A_k(M)} \cdot \left(< x^{j+k} > - < x^j > \cdot < x^k >\right) \tag{21}$$

Proof: First of all, it holds that:

$$\begin{aligned}\frac{\partial \prod_{j=1}^{M} A_j^{x_i^j}(M)}{\partial A_k(M)} &= x_i^k \cdot A_k^{x_i^k - 1}(M) \cdot \prod_{j \neq k, j=1}^{M} A_j^{x_i^j}(M) \\ &= \frac{x_i^k}{A_k(M)} \cdot \prod_{j=1}^{M} A_j^{x_i^j}(M)\end{aligned} \tag{22}$$

Additionally, from (7):

$$\begin{aligned}\frac{\partial p_i}{\partial A_k(M)} &= -\frac{\prod_{j=1}^{M} A_j^{x_i^j}(M) \cdot \left(\sum_{i=1}^{N}\left[\frac{\partial \prod_{j=1}^{M} A_j^{x_i^j}(M)}{\partial A_k(M)}\right]\right)}{\left(\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)\right)^2} \\ &+ \frac{\left[\frac{\partial \prod_{j=1}^{M} A_j^{x_i^j}(M)}{\partial A_k(M)}\right]}{\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)}\end{aligned} \tag{23}$$

Using equation (22), we replace the quantities in brackets as follows:

$$\begin{aligned}\frac{\partial p_i}{\partial A_k(M)} &= -\left[\frac{\prod_{j=1}^{M} A_j^{x_i^j}(M)}{\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)}\right] \\ &\times \frac{\sum_{i=1}^{N}\frac{x_i^k}{A_k(M)} \cdot \prod_{j=1}^{M} A_j^{x_i^j}(M)}{\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)} \qquad (24) \\ &+ \frac{x_i^k}{A_k(M)} \cdot \left[\frac{\prod_{j=1}^{M} A_j^{x_i^j}(M)}{\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)}\right] \qquad (25)\end{aligned}$$

Notice that the expressions in brackets constitute the definition of p_i from equation (7). Therefore:

$$\frac{\partial p_i}{\partial A_k(M)} = \frac{x_i^k \cdot p_i}{A_k(M)} - \frac{p_i\left(\sum_{i=1}^{N} x_i^k\left[\frac{\prod_{j=1}^{M} A_j^{x_i^j}(M)}{\sum_{i=1}^{N}\prod_{j=1}^{M} A_j^{x_i^j}(M)}\right]\right)}{A_k(M)} \tag{26}$$

Observe that, by reforming the last term of equation (25), the p_i definition has re-emerged. Thus:

$$\frac{\partial p_i}{\partial A_k(M)} = \frac{x_i^k}{A_k(M)} \cdot p_i - \frac{p_i}{A_k(M)} \cdot \left(\sum_{i=1}^{N} x_i^k \cdot p_i \right) = \frac{p_i}{A_k(M)} \cdot \left(x_i^k - < x^k > \right) \quad (27)$$

Finally, according to the definition of the statistical moments:

$$< x^j >= \sum_{i=1}^{N} x_i^j \cdot p_i \Longrightarrow \frac{\partial < x^j >}{\partial A_k(M)} = \sum_{i=1}^{N} x_i^j \cdot \left[\frac{\partial p_i}{\partial A_k(M)} \right] \quad (28)$$

Replacing the bracketed quantity with (27) produces:

$$\frac{\partial < x^j >}{\partial A_k(M)} = \frac{\left(\sum_{i=1}^{N} x_i^{j+k} p_i - \left(\sum_{i=1}^{N} x_i^j p_i \right) < x^k > \right)}{A_k(M)} \quad (29)$$

Finally:

$$\frac{\partial < x^j >}{\partial A_k(M)} = \frac{\left(< x^{j+k} > - < x^j >< x^k > \right)}{A_k(M)} \quad (QED) \quad (30)$$

■

REFERENCES

[1] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," *Wireless Netw.*, vol. 5, no. 3, pp. 171–182, 1999.

[2] P. Nicopolitidis, G. Papadimitriou, and A. Pomportsis, "Continuous flow wireless data broadcasting for high-speed environments," *IEEE Trans. Broadcast.*, vol. 55, no. 2, pp. 260–269, 2009.

[3] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 199–210, 1995.

[4] J. Xu, X. Tang, and W.-C. Lee, "Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation," *IEEE Trans. Parallel Distributed Syst.*, vol. 17, no. 1, pp. 3–14, 2006.

[5] C. Ngo and V. Li, "Centralized broadcast scheduling in packet radio networks via genetic-fix algorithms," *IEEE Trans. Commun.*, vol. 51, no. 9, pp. 1439–1441, 2003.

[6] P. Nicopolitidis, V. L. Kakali, G. I. Papadimitriou, and A. S. Pomportsis, "On performance improvement of wireless push systems via smart antennas," *IEEE Trans. Commun.*, accepted, 2011.

[7] S. Guiasu and A. Shenitzer, "The principle of maximum entropy," *Mathematical Intelligencer*, vol. 7, no. 1, pp. 42–48, 1985.

[8] J. Gecsei, *The Architecture of Videotex Systems*. Prentice-Hall, 1983.

[9] Jing Cai, T. Terada, T. Hara, and S. Nishio, "On a cooperation of broadcast scheduling and base station caching in the hybrid wireless broadcast environment," in *Proc. 2006 International Conf. Mobile Data Management*, p. 72.

[10] B. Zheng, X. Wu, X. Jin, and D. L. Lee, "TOSA: a near-optimal scheduling algorithm for multi-channel data broadcast," in *Proc. 2005 International Conf. Mobile Data Manag.*, pp. 29–37.

[11] S. Kim and S. H. Kang, "Scheduling data broadcast: an efficient cut-off point between periodic and on-demand data," *IEEE Commun. Lett.*, vol. 14, no. 12, pp. 1176–1178, 2010.

[12] C.-L. Hu and M.-S. Chen, "Online scheduling sequential objects with periodicity for dynamic information dissemination," *IEEE Trans. Knowledge Data Eng.*, vol. 21, no. 2, pp. 273–286, 2009.

[13] C. Liaskos, S. Petridou, and G. Papadimitriou, "Towards realizable, low-cost broadcast systems for dynamic environments," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, pp. 383–392, 2011.

[14] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.

[15] C. K. Liaskos and G. I. Papadimitriou, "Ultra lightweight adaptation processes for scheduling servers in push-based systems," in *Proc. 2010 IEEE Symp. Commun. Veh. Technol.*, pp. 1–6.

[16] C.-J. Su, L. Tassiulas, and V. J. Tsotras, "Broadcast scheduling for information distribution," *Wireless Netw. J.*, vol. 5, no. 2, pp. 137–147, 1999.

[17] J. Xu, D.-L. Lee, Q. Hu, and W.-C. Lee, "Data broadcast," in *Wiley Series on Parallel and Distributed Computing*, I. Stojmenović and A. Y. Zomaya, editors. John Wiley & Sons, Inc, 2002, pp. 243–265.

[18] V. L. Kakali, P. G. Sarigiannidis, G. I. Papadimitriou, and A. S. Pomportsis, "Speeding up the adaptation process in adaptive wireless push systems by applying spline interpolation technique," in *Proc. 2010 IEEE Mediterranean Electrotechnical Conf.*, pp. 1263–1268.

[19] E. Desurvire, *Classical and Quantum Information Theory*. Cambridge University Press, 2009.

[20] L. Pietronero, E. Tosatti, V. Tosatti, and A. Vespignani, "Explaining the uneven distribution of numbers in nature: the laws of Benford and Zipf," *Physica A: Statistical Mechanics Appl.*, vol. 293, no. 1-2, pp. 297–304, 2001.

[21] D. W. Marquardt, "An algorithm for least-squares estimation of non-linear parameters," *SIAM J. Appl. Mathematics*, vol. 11, no. 2, p. 431, 1963.

[22] K. Ueda and N. Yamashita, "On a global complexity bound of the Levenberg-Marquardt method," *J. Optimization Theory Appl.*, vol. 147, no. 3, pp. 443–453, 2010.

[23] H. D. Dykeman, M. H. Ammar, and J. W. Wong, "Scheduling algorithms for videotex systems under broadcast delivery," in *Proc. 1986 International Conf. Commun.*, pp. 1847–1851.

[24] D. Aksoy and M. Franklin, "R×W: a scheduling approach for large-scale on-demand data broadcast," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 846–860, 1999.

[25] M. A. Sharaf and P. Chrysanthis, "On-demand broadcast: new challenges and scheduling algorithms," in *Proc. 200 Hellenic Conf. Manag. Data*.

[26] D. Aksoy and M. Leung, "Pull vs push: a quantitative comparison for data broadcast," in *Proc. 2004 IEEE Global Telecommun. Conf.*, pp. 1464–1468.

[27] C. J. Su and L. Tassiulas, "Broadcast scheduling for information distribution," in *Proc. 1997 IEEE Conf. Comp. Commun.*, pp. 109–117.

[28] S. Glassman, "A caching relay for the World Wide Web," *Comp. Netw. ISDN Syst.*, vol. 27, no. 2, pp. 165–173, 1994.

[29] S. A. Krashakov, A. B. Teslyuk, and L. N. Shchur, "On the universality of rank distributions of website popularity," *Comput. Netw.*, vol. 50, no. 11, pp. 1769–1780, 2006.



C hristos Liaskos received the Diploma degree in electrical and computer engineering in 2004 and the M.S. degree in medical informatics in 2008 from the Aristotle University of Thessaloniki, Thessaloniki, Greece, where he is currently pursuing the Ph.D. degree in communication networks.



Prof. Papadimitriou Georgios (M'89, SM'02) received the Diploma and Ph.D. degrees in Computer Engineering from the University of Patras, Greece in 1989 and 1994 respectively. In 1997 he joined the faculty of the Department of Informatics, Aristotle University of Thessaloniki, Greece, where he is currently serving as an Associate Professor. His main research interests include optical networks and wireless networks. Prof. Papadimitriou is Associate Editor of five IEEE journals. He is co-author of three international books published by Wiley. He is author or coauthor of more than 100 journal and 100 conference papers. He is a Senior Member of the IEEE.