A Novel HMM-Based Learning Framework for Improving Dynamic Wireless Push Systems Performance

Vasiliki L. Kakali, Panagiotis G. Sarigiannidis, Georgios I. Papadimitriou*, and Andreas S. Pomportsis
Department of Informatics, Box 888, Aristotle University of Thessaloniki, 54124, Greece
*corresponding author's e-mail: gp@csd.auth.gr

Abstract—A new machine learning framework is introduced in this paper, based on Hidden Markov Model (HMM), designed to provide scheduling in dynamic wireless push systems. In realistic wireless systems, the clients' intentions change dynamically, hence a cognitive scheduling scheme is needed to estimate the desirability of the connected clients. The proposed scheduling scheme is enhanced with self-organized HMMs, supporting the network with an estimated expectation of the client intentions, since the system's environment characteristics alter dynamically and the base station (server side) has no a-priori knowledge of such changes. Compared to the original pure scheme, the proposed machine learning framework succeeds in predicting the clients' information desires and overcomes the limitation of the original static scheme, in terms of mean delay and system's efficiency.

Keywords— Hidden Markov models, machine learning, scheduling, wireless networks

1. Introduction

For applications that deal with traffic, weather information, sightseeing guidelines and news distribution, data broadcasting is an efficient way for delivering information over wireless networks. In such applications, the client demands are often overlapping. Thus, the broadcast of one information item will probably satisfy a large number of clients that have the same demand at the same time period. Among other architectures (pull and hybrid approaches), the push systems [1, 2, 3] seem to be the most promising, since they achieve high scalability and low complexity. Their operation involves a server being located onto a base station, which makes a priori assumptions about the demand probability of the information items and schedules its broadcasts according to these assumptions without any participation of the client side. This paper focuses on push systems.

The problem considering these systems lies on the fact that the server can only make assumptions about the client demands of the information items or simply consider that all the items have equal demand probability to avoid the starvation of some items due to faulty assumptions. Thus, the "pure" push systems [1, 2, 3] are unable to adapt to a priori unknown client demands. Other adaptive push systems use a learning automaton module, trying to adapt to environments with a priori unknown client demands [4].

The main disadvantage of the specific adaptive approaches lies in the determination of the optimized values regarding the operational parameters of the learning automaton, especially in environments under dynamic alterations of their characteristics. For example, two critical parameters affect the learning automaton's performance. The former one, known as the convergence speed parameter, determines the speed and accuracy of the adaptation process, while the latter one, known as lowest probability threshold, affects the broadcast frequency of the unpopular items. Apparently, both parameters should be suitably tuned according to the related environment so that undesirable phenomena are avoided, such as low performance, starvation, slow convergence and lack of adaptive nature. Hence, the usage of learning automata demands fine tuning which presupposes a priori known environments. However, in most realistic cases, the environment includes unknown and dynamic parameters. This pitfall should be addressed with a novel compact prediction scheme operating independently of the optimized system parameters.

Extending our previous efforts [5] to environments where the clients' intentions and geographic locations change dynamically, a novel learning framework for wireless push systems is proposed in this paper, called Predictive Push Scheme (PPS). The scheme is designed on the basis of machine learning realized by Hidden Markov Models (HMMs) and targets at estimating the clients' intentions in environments with dynamic characteristics. HMMs have been widely used for prediction, machine learning, cognitive systems or pattern recognition [6]. The introduced learning method is evaluated by a scheduling algorithm, which is applied in order to accommodate the forthcoming broadcasts in the context of a wireless push network. In fact, HMMs provide a broadcast schedule, estimating the most desirable information items in accordance with their demand probability. The PPS aims at reducing the mean time delay that a client experiences waiting to receive the desired information item. Concurrently, according to the conducted simulation experiments, it seems to improve the wireless broadcast system performance.

The remainder of the paper is organised as follows: Section 2 presents the original static wireless push system, Section 3 analyses the proposed learning framework and simulation results are presented in Section 4. Finally, Section 5 concludes the paper.

2. A Static Environment Scheduling Scheme

The method presented in [2], optimizes the system performance of a static environment scheduling scheme. The algorithm operates as follows: The server contains a database of K information items, and p' is the vector of the estimated probability demand per item. Assuming that t is the current time and $T(i)$ is the time when item i was last broadcast, for each broadcast, the server selects to transmit the item i that maximizes the cost function (objective function):

$$CF(i) = (t - T(i))^2 \frac{p_i'}{l_i}(1 + E(l_i))/(1 - E(l_i)),\ 1 \le i \le K \qquad (2.1)$$

where p_i' is the estimated demand probability for item i, l_i is the item's length, $E(l_i)$ is the probability that an item of length l_i is received with an unrecoverable error, $T(i)$ is initialized to -1 and if the maximum value of $CF(i)$ is given by more than one items, the algorithm selects one of them arbitrarily. Upon the broadcast of item i at time t, $T(i)$ is changed so that $T(i)=t$.

The main limitation of this method is its static orientation as it does not support a mechanism able to update the estimated probability vector p' at environments with dynamic changes at the clients' preferences. In the remainder of this paper this method will be referred as the "static push scheme".

3. The Predictive Push Scheme

3.1 The System Architecture

The main entities of the studied push system are the base station and the set of connected mobile clients. Specifically, the system consists of P clients gathered into N groups and the members of each group may demand M different information items. The clients' grouping is being held logically, without disorientating the broadcast manner of the system. The base station consists of one omni-directional antenna, while the server, located into the base station, maintains a database of $K= N\times M$ different items. The server utilizes a number of $N\times M$ HMMs in order to schedule the item broadcasts as described in subsection 3.3. The base station broadcasts the server's data items while satisfied clients respond ("vote") to the server's broadcasts. For the uplink communication, the Code Division Multiple Access (CDMA) technique has been chosen [7]. Broadcasts are organized into transmission frames. Each frame is repeated periodically, comprising of two phases, the round robin phase and the prediction phase. During the round robin phase the server side broadcasts all the items (of each group) once per time. The clients' feedbacks are collected after each item broadcast. Then, the prediction phase begins and the transmission items are selected according to the output of the predictor component. The duration of each phase is equal to $N\times M$ broadcasts for fairness reasons. The system architecture is depicted in Fig. 1.

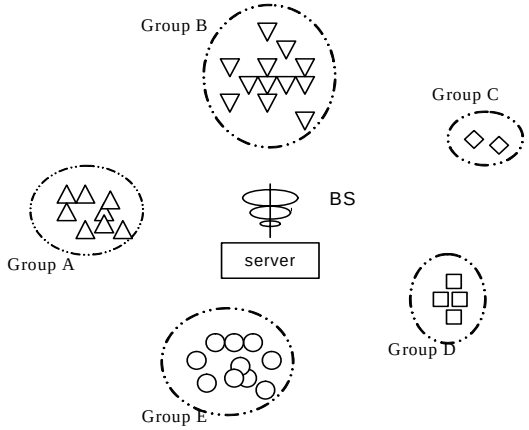

Fig.1. The system architecture.

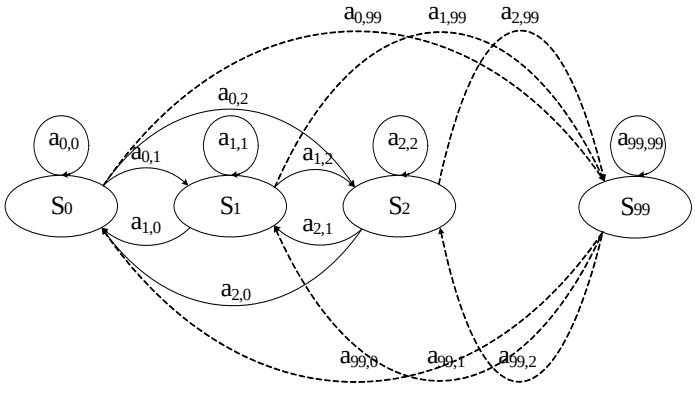

Fig.2. The Hidden Markov Model structure, defined by 100 individual states.

3.2 Preliminaries

According to [8], an HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. A process state is defined as a conditional distribution over the future – a measure space of semi-infinite sequences of symbols –

induced by conditioning on known historical information [9]. In other words, the process states are the possible states of knowledge of an observer who wishes to predict the future symbols with high accuracy.

Focusing on discrete time models, given that $t = 1, 2, \ldots$, denote time instants and that q^t is a random vector (of real values or symbols) then the entity $\{q^t\}$ is defined as a stochastic process. Consequently, supposing that q^t is a stochastic process taking values in S, which is either countable or finite having SN elements, a Markov chain is defined as follows:

$$P[q_t = S_j \mid q_{t-1} = S_i]$$

$$P[q_t = S_j \mid q_0 = S_u, ..., q_{t-2} = S_y, q_{t-1} = S_i] \qquad (3.2.1)$$

$$S_u, S_y, S_i, S_j \in S,\ 0 \le u, y, i, j \le SN-1$$

An HMM consists of a recurrent finite-state Markov Chain, an alphabet of output symbols, and a distribution over that alphabet for each transition in the Markov Chain. The states and transitions of the Markov Chain are hidden from observation so that only the output symbols are visible. More specifically, the elements of an HMM are: a) a finite number of states, b) a transition probability distribution which depends on the previous state and indicates a new state after each clock time t, and c) a set of observation probability distributions, which represent random variables or stochastic processes [8]. Formally, the following model notations stand for the designed HMM:

- SN stands for the number of states. For the following analysis an ergodic model is adopted, whereby all states are interconnected in such a way that any state can be reached from any other state [9].
- S denotes the set of individual states:
  $$S = \{S_0, \ldots, S_{SN-2}, S_{SN-1}\} \qquad (3.2.2)$$
- KN stands for the number of observation symbols, i.e., the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the designed system.
- K denotes the set of observation symbols per state:
  $$K = \{K_0, \ldots, K_{KN-2}, K_{KN-1}\} \qquad (3.2.3)$$
- A denotes the state transition probability distribution:
  $$A = \{a_{i,j}\}$$
  $$a_{i,j} = P[q_t = S_j \mid q_{t-1} = S_i] \qquad (3.2.4)$$
  $$0 \le i, j \le SN-1$$
  $$S_i, S_j \in S$$
  The state transition coefficients have the following properties:
  $$\sum_{i,j=0}^{SN} a_{i,j} = 1 \qquad (3.2.5)$$
  $$0 \le i, j \le SN-1$$
  Since the designed model is ergodic is holds:
  $$a_{i,j} > 0 \qquad (3.2.6)$$
  $$0 \le i, j \le SN-1$$
- B denotes the observation symbol probability distribution in state j:
  $$B = \{b_j(r)\}$$
  $$b_j(r) = P[K_r \text{ at time } t \mid q_t = S_j]$$
  $$0 \le j \le SN-1 \qquad (3.2.7)$$
  $$0 \le r \le KN-1$$
  $$K_r \in K$$
- π denotes the initial state distribution:
  $$\pi = \{\pi_i\},\ \pi_i = P(q_1 = S_i)$$
  $$0 \le i \le SN-1 \qquad (3.2.8)$$
  $$S_i \in S$$

3.3 The HMM scheme

It is well known that HMMs are very important due to the fact that they are able to realize noticeable practical systems –e.g., prediction systems, recognition systems, identification systems, stock market forecasting, etc, in a very efficient manner. Apparently, the server has no a priori knowledge concerning the dynamic client demands. In this perspective, novel learning framework is proposed to make the server capable of taking efficient prediction-based decisions. The proposed framework aims at estimating the requests of the connected clients. By having an estimated expectation of the client intentions, the server side is able to support more efficient item selections, resulting in decrement of the system response time.

The introduced system is composed of N groups, where each group member may request one of M possible items. Hence, $N \times M$ parallel HMMs are utilized, i.e., one machine learning module per group per item. Each one of these modules may be described at any time as being in one of a set of $SN = 100$ distinct states. Extending the Eq. 3.2.2, the set of individual states are given as follows:

$$S = \{S_0, \ldots, S_{98}, S_{99}\} \quad (3.3.1)$$

Each distinct state denotes an equivalent range of acceptance rate. The meaning of acceptance rate refers to the acceptance percentage of each broadcast based on the clients' feedbacks.

The server is able to possess an estimate of the number of clients that are under its coverage by sending one control packet asking of them to send back a feedback. Each client sends his feedback (vote) as an answer to the server's control packet. Then, the server, using its long-code database [7], decodes the votes possessing an estimation of client population. This procedure can be repeated periodically.

For example, consider a distributed system with 1000 connected clients, as described above. If the server side broadcasts item y of group x and the group x sends back 25 votes, the acceptance rate of group x for item y will be 2,5%.

The 100 acceptance states cover a range from 0-100% of the total client population (of the entire system). For instance, state S_0 indicates an acceptance range of 0-1% (0% is included) of the total client population. Similarly, state S_{50} indicates the acceptance range of 50-51% (50% is included). Lastly, state S_{99} represents the acceptance range of 99-100% (both included). Consequently, the observation symbols per state are as follows:

$$K = \{K_0, \ldots, K_{98}, K_{99}\} \quad (3.3.2)$$

Each symbol corresponds to each state and indicates the mean value of each acceptance range:

$$K_0 = 0.5\%, \ldots, K_{98} = 98.5\%, K_{99} = 99.5\%$$

The model designed is distributed and the Eq 3.2.1. is extended as follows:

$$\begin{aligned} & P[q_t^{x,y} = S_j \mid q_{t-1}^{x,y} = S_i] \\ & P[q_t^{x,y} = S_j \mid q_0^{x,y} = S_u, \ldots, q_{t-2}^{x,y} = S_y, q_{t-1}^{x,y} = S_i] \\ & S_u, S_y, S_i, S_j \in S,\ 0 \le u, y, i, j \le 99 \\ & 1 \le x \le N,\ 1 \le y \le M \end{aligned} \quad (3.3.3)$$

Obviously, $q_t^{x,y}$ denotes the process state at time t of the HMM corresponding to x group and y item. Hereafter, the state transition probability distribution is considered as follows:

$$\begin{aligned} & A^{x,y} = \{a_{i,j}^{x,y}\} \\ & a_{i,j}^{x,y} = P[q_t^{x,y} = S_j \mid q_{t-1}^{x,y} = S_i] \\ & 0 \le i, j \le 99 \\ & S_i, S_j \in S \\ & 1 \le x \le N \\ & 1 \le y \le M \end{aligned} \quad (3.3.4)$$

In this perspective, the state transition coefficients have the properties:

$$a_{i,j}^{x,y} > 0, \sum_{i,j=0}^{99} a_{i,j}^{x,y} = 1 \quad (3.3.5)$$

The HMM structure is depicted in Fig. 2. For example, transition probability $a_{15,23}^{3,2}$ corresponds to the transition from state S_{15} to S_{23}. This transition refers to the acceptance change from range of 15-16% to range of 23-24% of the total client population, concerning the third group and second item. In other words, the above transition indicates that the number of satisfied connected clients to the third group increased from 15-16% to 23-24%, regarding the broadcast of the second item.
The Eq. 3.2.7 is also extended in order to include the distributed nature of the model designed:

$$\begin{aligned} & B^{x,y} = \{b_j^{x,y}(r)\} \\ & b_j^{x,y}(r) = P[K_r \text{ at time } t \mid q_t^{x,y} = S_j] \\ & 0 \le j \le SN - 1 \\ & 0 \le r \le KN - 1 \\ & K_r \in K \\ & 1 \le x \le N \\ & 1 \le y \le M \end{aligned} \quad (3.3.6)$$

Finally, the initial state probabilities are the same for all groups and items and are denoted as follows:

$$\begin{aligned} & \pi = \{\pi_0\}, \pi_0 = P(q_1^{x,y} = S_0) \\ & S_0 \in S \\ & 1 \le x \le N \\ & 1 \le y \le M \end{aligned} \quad (3.3.7)$$

The learning framework tries to estimate the client requests in order to accommodate the forthcoming broadcasts. The introduced model, which is considered as memory-less since only the current and the predecessor state is known, needs a data structure to be able to support operations with memory demands. For this purpose, a set of history vectors are applied to store and handle the past states for each HMM. The number of entries of each history vector is a system parameter, denoted by *V*:

$$\begin{aligned} & H_z^{x,y}(h) = \{H_z^{x,y}(1), ..., H_z^{x,y}(V-1), H_z^{x,y}(V)\} \\ & 1 \le x \le N \\ & 1 \le y \le M \\ & 1 \le h \le V \\ & 0 \le z \le 99 \end{aligned} \quad (3.3.8)$$

In this manner, the entry $H_{90}^{3,2}(5) = 80$ means that the server has sent the second item and received votes from the third group, indicating the transition from state S_{90} to S_{80}, stored 5 entries before within the vector.

In essence, the proposed framework composes of a learning phase, during which the history vectors provide the model with a memory structure allowing it to know the current state amongst *V-1* past states. In practice, these vectors operate as first-in-first-out queues. The current (actual) state is stored in the first entry, while the value of the last entry is replaced by the penultimate one. All other values are shifted one position to the end of the vector.

History vectors work as follows: the actual clients' votes represent a state in each model and the number of state is stored in the vectors after each broadcast, keeping the model up-to-date. After the learning phase, a prediction phase follows, whereby the

model predicts the most probable state for each group and item, indicating the most probable acceptance rate for the next broadcast. For instance, suppose that after t^{th} broadcast the actual acceptance rate of group x for item y indicates ph_t state. Also, assume that after t'^{th} broadcast the corresponding state for the same group and item is $ph_{t'}$. Then it holds that the active current state of group x for item y after t^{th} broadcast is $q_t^{x,y} = S_{ph_t}$. Afterwards, the history vector update takes place as follows:

$$H_{ph_t}^{x,y}(h) = H_{ph_t}^{x,y}(h-1),$$
$$\text{for each } h, \text{ where } 2 \le h \le V \quad (3.3.9)$$
$$\text{and } H_{ph_t}^{x,y}(1) = ph_{t'}$$

Afterwards, the corresponding transition probabilities are updated, according to history vectors:

$$a_{ph_t,j}^{x,y} = \frac{frequency \;\; of \;\; j \;\; in \;\; H_{ph_t}^{x,y}}{V}$$
$$\text{for each } j, \qquad (3.3.10)$$
$$\text{where } 0 \le j \le 99$$

In the above equation, the frequency of *j* defines the number of times that state j appears in history vector $H_{ph_t}^{x,y}$.

For example, consider that after broadcast at time *t = 120* the actual acceptance rate of group 3, for item 2 indicates the state 40. Also assume that after broadcast at time *t' = 125* for the same group and item, the number of clients' feedbacks indicates state 51. Furthermore, let the size of the history vectors be equal to 100 entries. First, the update of the history vectors takes place: $H_{40}^{3,2}(h) = H_{40}^{3,2}(h-1)$, *for each h, where 2≤h≤100 and* $H_{40}^{3,2}(1) = 51$. Next the transition probabilities are changed, according to the history data of the vectors: $a_{40,j}^{3,2} = \frac{frequency \;\; of \;\; j \;\; in \;\; H_{40}^{3,2}}{100}$, *for each j, where 0≤j≤99.*

Finally, the predictor modules choose the most probable transition state for each item and store it to vector F_t, where t is the current time, as follows:

$$F_t(x,y) = \{F_t(1,1), F_t(1,2), ..., F_t(1,M),\; F_t(2,1), F_t(2,2), ...$$
$$F_t(2,M), ... F_t(N,1), F_t(N,2) ... F_t(N,M)\},$$
$$1 \le x \le N, 1 \le y \le M$$
$$S_{predicted} = \arg\max_{\zeta}[a_{ph_t,\zeta}^{x,y}]$$
$$F_t(x,y) = K_{S_{predicted}} \qquad (3.3.11)$$
$$1 \le x \le N$$
$$1 \le y \le M$$
$$ph_t, \zeta, S_{predicted} \in S$$
$$K_{S_{predicted}} \in K$$

In fact, vector F_t determines the most probable transition, given the current state for each group and item. As previously mentioned, the prediction phase follows the round robin phase. In this way, the former phase is used to fill the history vectors of the prediction modules, while the latter one produces the prediction-based schedule based on vector F_t. Therefore, vector F_t is activated only during the prediction phase. During this phase and after each broadcast, vector F_t is reformed in order to give the most probable transition of each group of each item per group. This fact means that vector F_t indicates the predicted acceptance rate of all $N \times M$ items (or M items per N groups). The next step is to apply a scheduling algorithm, which is utilized in order to construct a schedule of the forthcoming broadcasts based on the predicted acceptance rates. The server's operation order is depicted in Fig. 3.

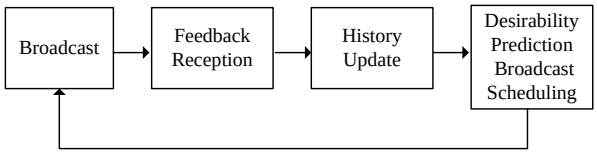

Fig. 3. The server's operation order.

The scheduling algorithm is defined as follows:

Scheduling Algorithm

//begin the prediction phase.
1. *At time t calculate F_t vector*
//a normalization takes place to schedule $N \times M$ (total number of items) broadcasts according to the output of the prediction modules
2. *Summarize the F_t vector and set it as* $Sum(F_t(x, y))$.
//Get the proportional acceptance rate (vector R_t) of each x group and each y item based on F_t vector:
3. *Set* $R_t(x, y) = \dfrac{F_t(x, y) \cdot P}{Sum(F_t(x, y))} \cdot N \cdot M$,
for each x, y, where $1 \le x \le N$, $1 \le y \le M$
//The selection of the forthcoming items takes place.
4. *While (* $Sum(R_t(x, y)) > 0$ *)*
5. *Find* $R\max_t(x\max, y\max) = \max[R_t(x, y)]$ *and select item ymax of group xmax, as the next item to broadcast.*
6. *Broadcast item ymax of group xmax.*
7. *Reduce R_t(xmax, ymax) by one.*
//history vectors update
8. *end_while*
//end of schedule algorithm

It is worth mentioning that the proposed scheme does not induce extra computational time. More specifically, the proposed scheme as well as the one of [2] runs linearly in sense of the number of groups and items. The complexity of scheme [2] is bounded by O($N \cdot M$) while the proposed one runs in O($N \cdot M \cdot SN \cdot (V+1)$). Since *SN* and *V* are constant and predefined, both schemes cause the same computational time.

4. Performance Evaluation

4.1 Simulation Environment

The server contains a database of $K=N \times M$ equally-sized items, with item length *l*, being equal to the unit. A population of *P* clients is considered, grouped into *N* groups. Each group is located at a different region and has different item demands. In order to model groups of clients with different group sizes, the size of each group is computed via the Zipf distribution. Thus, the number of clients in group *x*, *Size(x),* $1 \le x \le N$, is:

$$Size(x) = c\left(\frac{1}{x}\right)^{\theta} \cdot P \qquad (4.1.1)$$

where $c = \dfrac{1}{\sum_k \left(\frac{1}{k}\right)^{\theta}}$, $k \in [1..N]$ and θ is a parameter named group size skew coefficient. Any client belonging to group *x* is interested in the same subset B_x of server's data items. All items outside this subset have a zero demand probability at the clients of the group. Moreover, $B_{x1} \neq B_{x2}, \forall x1, x2 \in [1..N]$, $x1 \neq x2$, which means that there are not any common demands between any two clients belonging to different groups.

In each subset, the client demand probability p_y for each item in place *y* in that subset, is computed also via the Zipf distribution with θ_1 being the item demand skew coefficient.

As the number of items per subset B_x is considered the same for each subset (equal to $M=K/N$), a number of *NxM* parallels HMMs are utilized at the server, as they are described in section 3.

The broadcasts are subject to reception errors, with unrecoverable errors per instance of an item occurring, according to a Poisson process with rate λ. The simulation environments are dynamic with either parameter θ (group size skew coefficient) or θ_1 (item demand skew coefficient) altering its value in the interval of [0.0…1.0] and [0.0…1.5], respectively, randomly during the simulation. The simulation runs until the server broadcasts *BR* items. The PPS is compared with the "static push system" of [2] in environments with dynamic characteristics. According to "static push system", the item demand probability vector is uniformly distributed, which means that the server assumes that each item is equally desired by the clients. Eventually, this vector remains steady during the whole simulation. The comparison of the PPS with the adaptive scheme can not be performed as the operational parameters of the learning automaton depend dramatically on the environment nature and differentiate between the examined scenarios. These values can not be pre-estimated due to the a priori unknown and dynamic altering characteristics of the examined scenarios that appear in the majority of the realistic cases.

The performance of the compared schemes is measured in terms of mean delay and efficiency. The mean delay is defined as the amount of time (measured in number of broadcasts) a client has to wait for the item that he needs, while the system efficiency is defined as the efficiency of the broadcast process. More specifically, the system utilization per client is defined as follows: $Effic_g = \frac{number\ of\ satisfied\ broadcasts}{total\ number\ of\ broadcasts}$, where $1 \le g \le P$. In this manner, the utilization of the entire system is given by $\frac{\sum_{g=1}^{P} Effic_g}{P}$.

4.2 Simulation Results

The experiments are performed in a simulator coded in Matlab. The simulation results presented in this Section are obtained with the following values to the parameters: $P = 1000$, $BR = 100000$, $\lambda = 0.1$ and $V = K$. Table 1 summarizes the characteristics of the networks that have been simulated. Figures 4-13 depict the simulation results of these networks. The number of simulation runs for each dotted data point is equal to 10 and the confidential interval is 95%.

Table1. The characteristics of the network environments.

Network	N	K	θ	θ_1
N_1	1	50…300	-	dynamic change
N_2	5	250	0.0 ... 1.0	dynamic change
N_3	5	250	dynamic change	0.0 ... 1.5
N_4	5	50…500	dynamic change	0.8
N_5	1…10	500	dynamic change	0.9

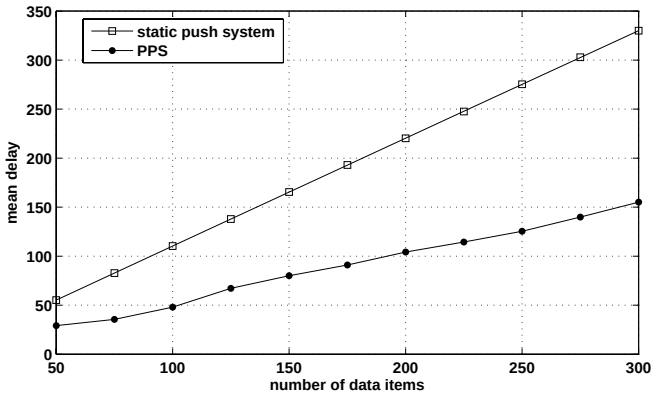

Fig. 4. Mean delay of network N_1, where the number of database items varies and the item demand skew coefficient changes dynamically during the simulation.

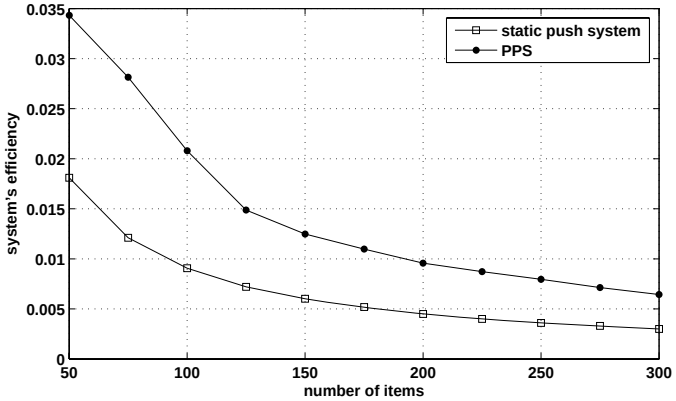

Fig. 5. The efficiency of network N_1, where the number of database items varies and the item demand skew coefficient changes dynamically during the simulation.

Figures 4 and 5 present the mean delay and the system efficiency respectively versus different values of number of database items *K*. In network, N_1, the item demand skew coefficient θ_1 varies dynamically during the simulation, taking values in the interval of [0.0…1.5], expressing the client desirability changes in realistic broadcasting environments. For these experiments, the client population is considered to belong to a single group ($N = 1$), while the number of data items changes taking values from 50 to 300 with step equal to 25. Figures 4 and 5 confirm that the proposed PPS scheme achieves lower mean delay compared to the "static push" one, while the system efficiency of the PPS scheme is higher than the "static push" one for various values of the *K* parameter. The applied HMM component seems to be beneficial to the proposed framework, by supporting accurate estimations regarding the most desirable information items. In this manner, the enhanced push system is able to take more effective schedule decisions compared to the "static push" one. It is clear that as the number of data items increases, the system mean delay increases too. This behaviour lies in the fact that in a system where the clients have to choose from a small pool of options (e.g., K = 50), a single broadcast is able to satisfy a large number of clients, compared to a system where the number of the available options is much higher (e.g., K = 300). In other words, when the number of the available data items increases, the differentiation in clients' desirability is high. Thus, the server needs more time to satisfy the whole client population. As an effect, the system mean delay increases and the system efficiency degrades.

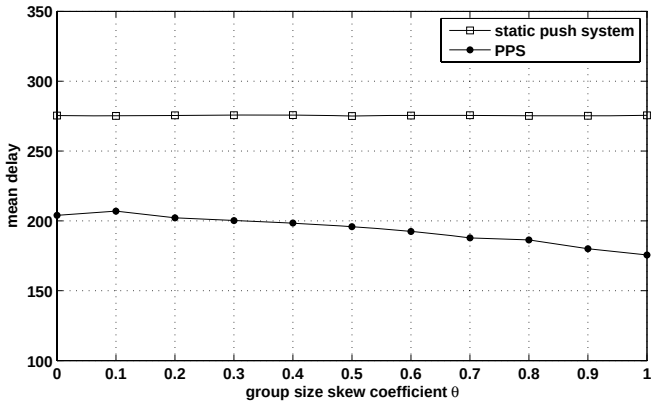

Fig. 6. Mean delay of network N_2, where the group size skew coefficient varies and the item demand skew coefficient changes dynamically during the simulation.

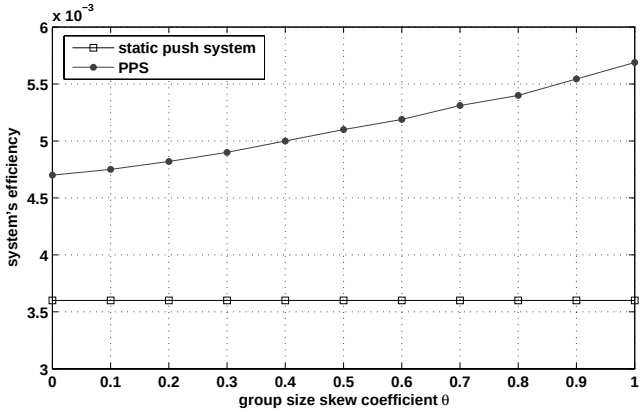

Fig. 7. The efficiency of network N_2, where the group size skew coefficient varies and the item demand skew coefficient changes dynamically during the simulation.

The performance of network N_2 is evaluated in figures 6 and 7, as the skew coefficient θ varies. In this network, the clients are separated into five groups (G = 5), the number of data items is stable and equal to 250, while the item demand skew coefficient θ_1 takes various values in the interval of [0.0…1.5] (dynamic environment). It is obvious that the PPS scheme not only exceeds the performance of the "static push scheme" (lower mean delay) but also improves its performance, in terms of system efficiency, as the variation between the group sizes increases (the value of θ increases). The "static push" scheme is unable to manipulate the alterations of the items' desirability (expressed by the value of θ_1) as well as the variation between the group sizes. Thus, it fails to follow the dynamic features of a realistic environment, keeping a stationary performance. On the contrary, the PPS scheme is able to take into account not only the alterations of the items' desirability, but also the variation of groups' size through the presence of the designed prediction-based framework. Thus, the PPS achieves further performance improvement in networks with high value of group size skew coefficient θ, as a single broadcast succeeds to satisfy a large number of clients.

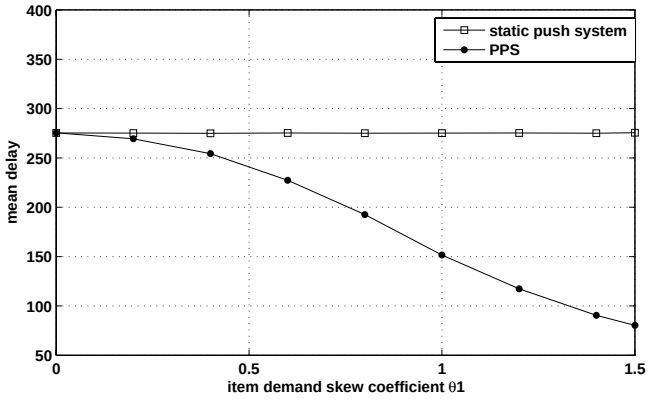

Fig. 8. Mean delay of network N_3, where the item demand skew coefficient varies and group size skew coefficient changes dynamically during the simulation.

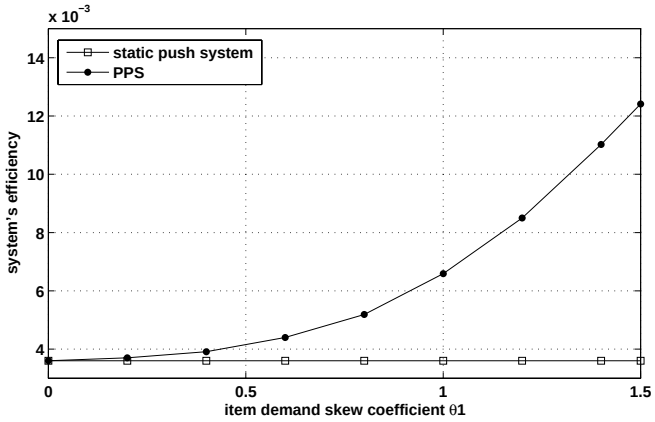

Fig.9. The efficiency of network N_3, where the item demand skew coefficient varies and group size skew coefficient changes dynamically during the simulation.

In figures 8 and 9, the mean delay and system efficiency of network N_3 are depicted respectively concerning the coefficient θ_1. In this network, the number of groups is set to 5 ($G = 5$), while the number of data items is set to 250 ($K = 250$). Furthermore, the client demands are formed with the parameter of θ_1, varying from 0.0 to 1.5. For this scenario, a dynamic change of group size skew coefficient θ, takes place, reflecting the size of the system groups. Once more, the suggested framework reduces the waiting time of the clients compared to the "static push" one. Also, the proposed scheme outperforms the static one in terms of system efficiency, presenting efficient and effective behaviour, as the item demand skew coefficient θ_1 raises. The PPS scheme achieves better performance than the static one, since it is able to predict the client demands of the system. Moreover, a high value of the skew coefficient θ_1 causes reduction in the number of the items that are the most popular. Thus, the broadcast of one of these highly desirable items is likely to satisfy the majority of the clients. In this way, the system experiences lower mean delay and higher system performance.

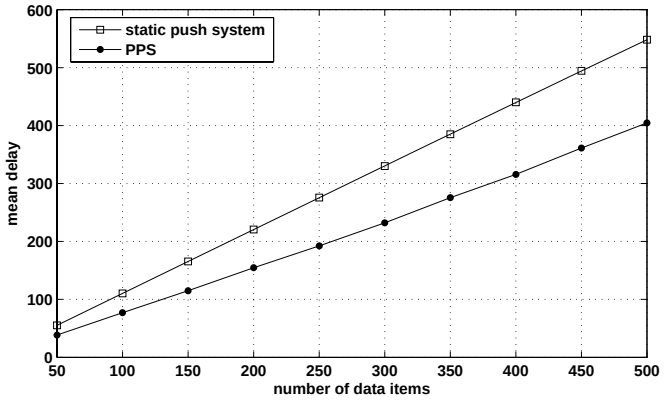

Fig. 10. Mean delay of network N_4, where the number of database items varies and the group size skew coefficient changes dynamically during the simulation.

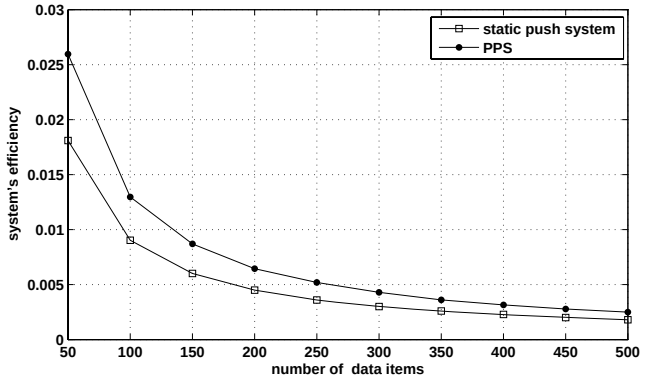

Fig. 11. The efficiency of network N_4, where the number of database items varies and the group size skew coefficient changes dynamically during the simulation.

The network scenario N_4 is evaluated in figures 10 and 11, in terms of mean delay and system efficiency, respectively, against the number of database items. The environment consists of five groups (G = 5), while the number of database items varies in the interval of [50… 500]. The group size skew coefficient θ alters dynamically during the simulation, while the item

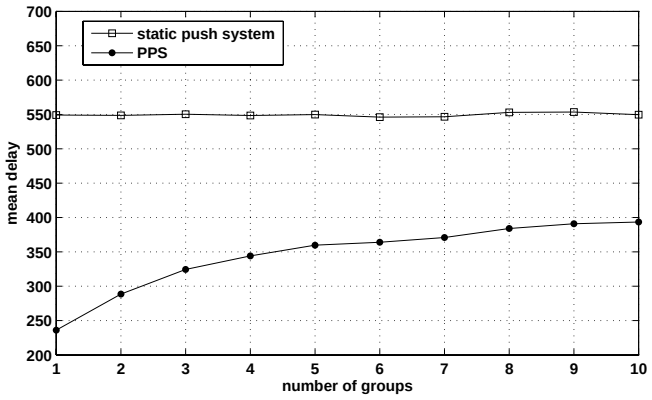

Fig. 12. Mean delay of network N_5, where the number of groups varies and the group size skew coefficient changes dynamically during the simulation.

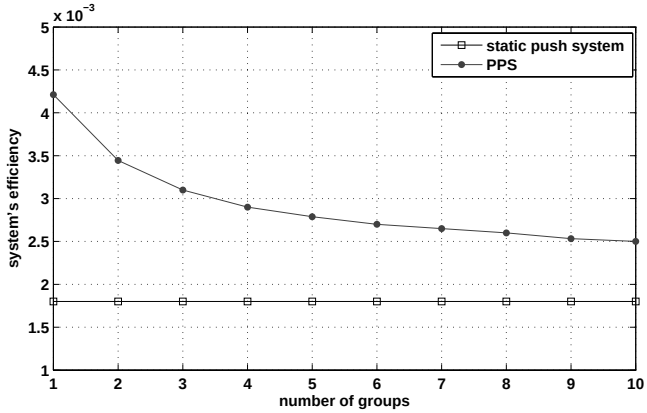

Fig. 13. The efficiency of network N_5, where the number of groups varies and the group size skew coefficient changes dynamically during the simulation.

demand skew coefficient θ_1 is constant and equal to 0.8. Clearly, the mean delay is almost proportional to the number of the items and the efficiency of the corresponding network decreases as the number of items increases. The proposed PPS scheme excels in both figures, keeping the mean delay low, while the broadcasts are more effective.

Figures 12 and 13 confirm the superiority of the proposed scheme, since it is apparent that it excels the "static push" one, in terms of mean delay and system efficiency for environments with different number of groups of clients (N=1…10).

Totally, the following observations can be extracted from the presented simulation results:

1) The proposed PPS scheme succeeds in predicting client demands and it is clearly superior to the "static push" one, since it provides the desired items to the connected clients in shorter time than the "static push" one. In this manner, a client waits less amount of time for a desired item, compared to time induced by the "static push" operation. Hence, by utilizing the proposed scheme more clients receive the desired items in less time.

2) The suggested framework reduces significantly the mean delay irrespectively the number of groups and items. The prediction module provides the system with the ability to estimate and accommodate the appropriate broadcasts in order to service clients in expeditious manner.

3) The suggested prediction-based enhancement is capable of adjusting its operational attitude in accordance to the dynamic changes of a realistic environment.

Overall, the proposed prediction add-on enables a rigorous client intention handling, offering a more effective exploitation of the systems resources.

5. Conclusion

A novel learning framework was presented in this paper for dynamic wireless push systems. The applied scheme involves Hidden Markov Models (HMMs) and aims at supporting the system with accurate predictions regarding the selection of the most desirable client demands, in order to operate efficiently in dynamic environments. The dynamic alterations of the environment characteristics may involve changes of the items' desirability and variations of the groups' size that the clients are gathered into. The proposed framework estimates the forthcoming clients' intentions, based on the history data structure of the HMMs. Then, the outcome of the HMM framework is used to conduct a schedule of the forthcoming server's broadcasts that will serve the clients in the most efficient way. The simulation results reveal that the proposed scheme presents better performance than the "static push" one, resulting in reduced mean delay and higher system's efficiency.

References

[1] S. Acharya, M Franklin, and S.Zdonik, "Dissemination-based data delivery using broadcast disks", IEEE Pers. Commun., vol. 2, no. 6, pp.50-60, Dec. 1995.

[2] N.H.Vaidya and S.Hameed, "Scheduling Data Broadcast In Asymmetric Communication Environments", Wireless Networks, vol. 5, no.3, pp.171-182, May 1999.

[3] E.Yajima, T. Hara, M. Tsukamoto, S. Nishio, **"**Scheduling and caching strategies for correlated data in push-based information systems**",** ACM SIGAPP Applied Computing Review, vol. 9, no. 1, pp. 22-28, 2001.

[4] P. Nicopolitidis, G. I. Papadimitriou and A. S. Pomportsis, " Using Learning Automata for Adaptive Push - Based Data Broadcasting in Asymmetric Wireless Environments ", IEEE Transactions on Vehicular Technology, vol. 51, no. 6, pp. 1652-1660, Nov. 2002.

[5] V. Kakali, P. G. Sarigianidis, G. I. Papadimitriou, A. S. Pomportsis, "A prediction-based scheme for wireless push systems, using a statistical hidden Markov model", in Proceedings of 15th IEEE/SCVT (Symposium on Communications and Vehicular Technology), pp. 125-129, Nov. 2008.

[6] L.R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition”, Proceedings of IEEE, vol. 77, no. 2, pp. 257-286, Feb. 1989.
[7] W. C. Y. Lee, “Overview of Cellular CDMA”, IEEE Trans. on Vehicular Technology, vol. 40, no. 2, pp. 291-302, May 1991.
[8] L. Rabiner, B. Juang, “An introduction to hidden Markov models”, IEEE *ASSP Magazine,* vol. 3, no. 1, pp. 4-16, Jan 1986.
[9] G. Yin, Q. Zhang, “Discrete-time Markov chains: Two time-scale methods and applications”, Springer, New York, 2005.