Send Orders for Reprints to reprints@benthamscience.ae

A QoS-aware Routing for Multi-Services in Wireless Sensor Networks

Haifeng Lin* and Anna Jiang

College of Information Science and Technology, Nanjing Forestry University, Nanjing, 210037, China

Abstract: In this paper we propose a new routing algorithm for emergency bound based on multiple services for wireless sensor networks. In this routing algorithm channel conditions are associated with each time unit to provide overall fairness and maximize the total throughput of wireless sensor network. The proposed system model supports multiple users of each sensor, each user may have several service demands, every demand has different characteristics such as emergency bound, etc. How to guarantee the fairness for each user and the emergency bound for each service in wireless sensor network is the problem to be solved. In addition, we investigate the channel condition for each user and provide the maximum throughput for the single channel of each time unit.

Keywords: Wireless sensor networks, optimal scheduling, long fairness, shot fairness.

1. INTRODUCTION

Wireless sensor networks (WSNs) are popular due to their potential applications in civil and military domains [1, 2]. WSNs consist of many devices which can process data, store data and communicate with others via short range radio connections. They have no infrastructure, but they can configure themselves as a network and construct the routing table. New Quality of Service (QoS) models and algorithms are developed to evolve wireless sensor network into a true integrated services network.

However, where needs our most attention of wireless sensor networks today is that delay bound, fairness and throughput are crucial factors of QoS [1-5]. Service providers may have some constant bandwidth within the channel. At any time instance of each sensor, *m* users may initiate *n* services, here we assume *n* is a constant (which is the maximum service number that a channel can provide), and *m* is a variable. Consider also that the channel is a variable. Since there are so many changing variables, we need to investigate how to guarantee the delay bound for every service, fairness for every user and maximum throughput available [6-7]. This process is complicated to solve and is NP-complete.

Problem statement: suppose we have a single channel within the base station, its capacity *E* is a constant. It can contain *m* users and *n* services (maximum). This represents the ideal case. Now, the channel condition and users are changing instantaneously. We will use the stochastic process to simulate the channel condition and the users. Our objective is to provide the delay bound for each service of each user, and also guarantee the fairness of each user and maximize it throughput for each time unit – (transaction time for the channel). Delay bound, fairness and throughput are crucial factors of QoS [8-11]. If we satisfy this, both service providers and users will be satisfied, which is our purpose and contribution.

From the above, delay bound, fairness and throughput are three critical factors within QoS. However, some previous papers did not address them together: they addressed several facets under the limited conditions such as short-term periods, unique service provider, etc [7-14]. In this paper, we will present a fairly-practical model: multiple services, multiple users, and for the delay bound scheduling, we only consider those flows with good channel quality. We are not considering those flows with bad channel qualities, but their shares used by other flows should be compensated once their channel quality is good. We will consider both long-term and short-term fairness. Since each time unit we just allocate the limited bandwidth to the sessions with good channel quality, the overall throughput can be maximized. This is our contribution.

The rest of this paper is organized as follows. In section 2 we describe the system model that we are assuming and in section 3, we formulate the problem of scheduling in the wireless system, and also propose our new algorithm, with some derived formulas and pseudo codes involved. In section 4, some proof will be given. Next in section 5, we present simulation results. Finally, we state conclusion in section 6.

2. SYSTEM MODEL (FRAMEWORK)

Here we present Fig. (**1**), followed by some explanations.

$\varphi_i^j(S_0)$: Minimum bandwidth for user i, service j and session S at time t = 0;

$C_i^j(t)$: Channel condition of user i, service j at time t;

1874-4443/14 2014 Bentham Open

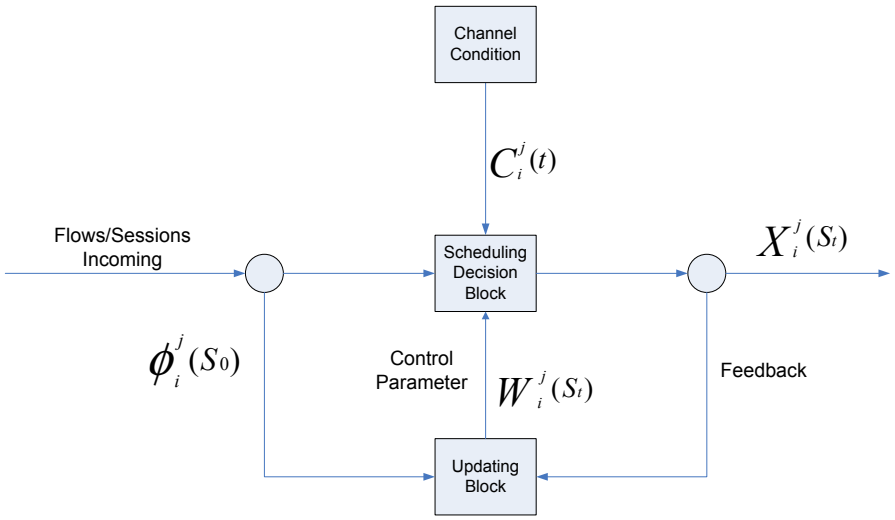

Fig. (1). System Model.

$W_i^j(S_t)$: Control parameter of user i, service j and session S at time t;

$X_i^j(S_t)$: Actual channel bandwidth usage for user i, service j and session S at time *t*;

3. ALGORITHM (PSEUDO CODE)

2 queues (lagging queue and leading queue)

Some compensation mechanisms

($L_i^j(S_t)$, $D_i^j(S_t)$) (L: queue length, D: delay bound, i: user #, j: service #, S: Session #, t: time t), L and D are changing each time unit.

Note: D: delay bound which is a constant associating with the incoming queue length and service type; d: emergency bound which is a variable, which changes every time unit based on the channel transaction time (here, we assume 1s). D = d when t = 0.

S0: The time when the session S is submitted to the channel;

$\varphi_i^j(S_0)$: Minimum bandwidth allocated to session S when t = 0;

$\varphi_i^j(S_0) = W_i^j(S_0) = l_i^j(S_0)/(l_i^j(S_0)/d_i^j(S_0)) = d_i^j(S_0)$ (here, d = D since t = 0);

……

if ($\varphi_i^j(S_0)$ + lagging (St-1) < $l_i^j(S_0)$)

/* for the lagging sessions */

{

$W_i^j(S_t) = \varphi_i^j(S_0)$ + lagging (St-1);

S0 = St;

$l_i^j(S_0) = l_i^j(S_0) - W_i^j(S_t)$;

}

else if (lagging (St-1) > $l_i^j(S_0)$)

{

$W_i^j(S_t) = l_i^j(S_0)$;

dequeue;

}

or

if ($\varphi_i^j(S_0)$ - leading (St-1) > = 0)

/* for the leading sessions */

{

$W_i^j(S_t) = 0$;

lagging (St) = $\varphi_i^j(S_0)$ - leading (St-1);

}

else

{

$W_i^j(S_t) = 0$;

leading (St) = leading (St) - $\varphi_i^j(S_0)$;

}

refresh lagging queue and leading queue;

Here, we assume the channel condition is good for both lagging and leading sessions at time t.

lagging (St) = leading (St) ($d \in [0, \infty]$). (all the time, the sum of the bandwidth of lagging queue should be the same as the sum of the bandwidth of the leading queue).

User simulation: generate a random integer number $U(t)$ each time unit. $U(t) \in [0, Umax]$. $Umax = E / min\, d^j$ ($j = 1..N$).

Guarantee delay bound: we only consider those flows with good channel quality. For those flows with bad channel quality, there is no need to consider them. But their shares used by other flows should be compensated once their channel quality is good. For example: if one service, its delay bound is 3pkgs/s, if its queue length is 12 packets, then all the packets must be sent within 12/3 = 4s; by using this algorithm, $\varphi_i^j(S_0) = W_i^j(S_0)$ = 12 / 4 = 3pkgs/s; if for some reason (limited bandwidth), in the first second, we can only allocate 1 packet bandwidth to this session, then we must do some compensation work once this session is in good channel quality again. So in this case, $W_i^j(S_t)$= 3 + 2 / 3 = 11 / 3pkgs/s, that is, we must allocate more bandwidth to this session to guarantee its delay bound.

Fairness: includes long term fairness and short term fairness.

Long term fairness: During a busy period long enough, if a session becomes error-free (good channel condition), as long as there is enough service demand, it should get back all the services "lost" while it was in error (bad channel condition).

Thus, a session which becomes error-free will eventually get back its entire "lost" services.

Short term fairness: The difference between the normalized services received by any two error-free sessions that are continuously backlogged and are in the same state during a time interval should be bounded.

This short term property is a generalization of the well-known fairness property in classical PFQ algorithms. The requirement that sessions in the same state receive the same amount of normalized service implies that (1) leading sessions should be penalized by the same normalized amount during compensation, (2) compensation services should be distributed in proportion to the lagging session's rate, and (3) when services from error sessions are available, lagging sessions receive these services at the same normalized rate, so do the leading sessions.

Maximize the channel throughput: in this model, we allocated the channel bandwidth to the session only with good channel quality, which means all the bandwidth will 100% be used, with no lost packets. So the channel throughput will be maximized throughout the time.

4. SOME PROOFS

Lemma 1 The lag of an error-free session is never greater than Lmax, where Lmax represents the maximum size of a message.

Proof. The proof is by induction. From the above algorithm, the lag of an error-free session i changes in one of the three following cases: (a) session i becomes active, (b) session i is selected but since it is leading another session j is selected to receive service, and (c) session i receives service from another session j.

Basic step. When an error-free session i becomes active, its lag is set to zero, and therefore the lemma is trivially true.

Inducting step. Assume lagi ≤ Lmax. We consider two cases: (1) lagi < 0, and (2) 0 ≤ lagi ≤ Lmax. Since in case (1) session i is leading, its lag can increase only when its service is given to another session j (see case (b) above). In this case, we have

$$\text{lagi} = \text{lagi} + l_j^k \le l_j^k \le \text{Lmax}; \quad (1)$$

where l_j^k represents the length of the packet at the head of the queue of session j. In case (2), session i is non-leading, and so its lag can only decrease (case (c) above). Thus, the bound holds.

Theorem 1 The difference between the normalized service received by any two sessions i and j during an interval [t1, t2) in which both sessions are continuously backlogged, error-free, and their status does not change is bounded as follows:

We consider two cases: both sessions are (1) lagging, or (2) leading during the entire time interval[t1, t2).

$$|Wi(t1, t2) - Wj(t1, t2)| \le \text{dmax}\ (t2 - t1) \quad (2)$$

for 2 lagging sessions;

where Wi(t1, t2) represents the service received by session i during [t1, t2), dmax is the maximum delay bound.

Proof. (1) Suppose both sessions are lagging during the entire interval [t1, t2). In this case both sessions receive service each time they are selected, or when they receive compensation from a leading session. So it follows that the total service received by an error-free lagging session during [t1, t2) is bounded by

$$0 \le Wi(t1, t2) \le di(t2 - t1) \quad (3)$$

similarly

$$0 \le Wj(t1, t2) \le dj(t2 - t1) \quad (4)$$

thus, from the above inequality, it is easy to see that

$$|Wi(t1, t2) - Wj(t1, t2)| \le \text{dmax}\ (t2 - t1) \quad (5)$$

or

$$\alpha * i - \beta * j \le (E - di - dj) * ((di - dj) / (di + dj)) \quad (6)$$

for 2 leading sessions;

In the case of α, β are the rate for two leading sessions, E is defining as channel capacity and therefore, we suppose di > dj.

Proof. (2) Suppose both sessions are leading at time t. And at time t, we have only these two sessions, in order to keep assigning bandwidth proportionally, the actual

Table 1. The Definition of Service.

ID	Service Name	Delay Bound
0	Audio	10pkgs/s
1	Video	20pkgs/s
2	FTP	1pkg/s
3	Http	5pkgs/s

bandwidth assigned to session i is α* i = (E - di - dj) * (di / (di + dj)), and the actual bandwidth assigned to session j is β* j = (E - di - dj) * (dj / (di + dj)), so it is easy to see that

$$\alpha i - \beta * j \leq (E - di - dj) * ((di - dj) / (di + dj)) \quad (7)$$

at time *t*.

or

$$\alpha * i - \beta * j \leq (E - di - dj) * ((di - dj) / (di + dj)) * (t2 - t1) \quad (8)$$

during the entire interval[*t*1, *t*2).

Theorem 1 says that the difference between the normalized services received by two error-free active sessions during any time interval in which they are in the same state is bounded (i.e., leading or lagging is bounded) (short-term fairness).

Theorem 2 Consider that an active lagging session i becomes error-free after time t. If session i is continuously backlogged after time t, then it is guaranteed to catch up after at most Δ units of time,

$$\Delta = \{ t+1, \text{ if t} \geq \frac{l_0}{d_0} \textbf{ or } \frac{l_0}{d_0}, \text{ otherwise } \} \quad (9)$$

Where l_0 is the packet length in time 0, and d_0 is the delay bound in time 0.

Theorem 2 says that the time it takes a lagging session that no longer experiences errors to catch up is bounded (long-term fairness).

Proof. From the pseudo code above, we can clearly see the following result:

if ($\varphi_i^j(S_0)$ + lagging (St-1) < $l_i^j(S_0)$)

/* for the lagging sessions */

{

$W_i^j(S_t) = \varphi_i^j(S_0)$ + lagging (St-1);

S0 = St;

$l_i^j(S_0) = l_i^j(S_0) - W_i^j(S_t)$;

} /* this section gives the upper bound of $\frac{l_0}{d_0}$ */

else if (lagging (St-1) > $l_i^j(S_0)$)

{

$W_i^j(S_t) = l_i^j(S_0)$;

dequeue;

} /* this section gives the upper bound of t+1 */

5. SIMULATION EXPERIMENTS

5.1. Service Definition

Here we define n services (n = 4) which can be accepted by the channel, along with the delay bound of each service, see Table **1**:

5.2 Session Generator

Here we present the C++ code to generate the sessions randomly, along with the session ID and session length, see the following code:

```
/* Use this code to generate the sessions. */
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
 srand(time(0));
 int S[4];
 int L[4];
 for(int i=0;i<4;i++){
S[i]=rand()%4;
L[i]=rand()%100+1;
 }
 for(int j=0;j<4;j++){
cout<<"S["<<j<<"]="<<S[j]<<" ";
cout<<"L["<<j<<"]="<<L[j]<<" ";
cout<<endl;
 }
```

```
return 0;
}
```

Note: array S is the service ID (0-3), array L is the session length of each service (1-100), and the results are similar to the following:

S[0] = 3	L[0] = 84
S[1] = 1	L[1] = 18
S[2] = 2	L[2] = 61
S[3] = 2	L[3] = 23

Note: from the above table, we can clearly see that we have generated 4 sessions at the same time. For example, the ID of session 0 is 3, it means this is the http service, and the session length is 84 packets, since the delay bound of http service is 5pkgs/s, so we must guarantee this session should be received by $\lceil 84/5 \rceil$ = 17s to keep the delay bound, etc.

5.3. White Noise Generator

At this time, we only focus on the scheduling algorithm itself rather than the physical channel. Here we use a Gaussian random variable generator to generate the channel noise. Later on, more complex channel models will be introduced.

Following is the code implemented by C++ to simulate the channel condition (C++ Gaussian noise generation).

```
/* Use this code to generate the white noise per second, it conforms to the Gaussian distribution. */
#include <cstdlib>
#include <iostream>
#include <ctime>
using namespace std;
const int numSamples=3;
int main()
{
 srand(time(0));
 const static int q=15;
 const static float c1=(1<<q)-1;
 const static float c2=((int)(c1/3))+1;
 const static float c3=1.f/c1;
 float random=0.f;
 float noise=0.f;
 for(int j=0;j<10;j++){
    cout<<"time:"<<j<<endl;
    for(int i=0;i<=numSamples;i++){
 random=((float)rand()/(float)(RAND_MAX+1));
noise=(2.f*((random*c2)+(random*c2)+(random*c2))-3.f*(c2-1.f))*c3;
 cout<<noise<<endl;
}
sleep(1);
}
 return 0;
}
```

Note: Gaussian random numbers. This algorithm implements a generation method for Gaussian distributed random numbers with mean = 0 and variance = 1 (standard Gaussian distribution) mapped to the range of -1 to 1 with the maximum at 0.The q variable defines the precision, with q = 15 the smallest distance between two numbers will be 1/(2^q div 3)=1/10922 which usually gives good results. And the results are similar to the following ('sleep (1)' is to guarantee that we generate the value to simulate the channel condition for each session per second, if the value is greater than 0, that means the channel condition for this session is good, or if the value is not greater than 0, that means the channel condition for this session is bad):

5.4 Simulation Results

Here we present the following table of the simulation results according to the assumptions above (the results based on step 2 – session generator and step 3 – white noise generator), in the next section, we will analyze how to get these results and why it can guarantee the 3 properties (delay bound, fairness, throughput) we have proofed before:

5.5. Results Analysis

From the tables above, we can clearly guarantee the delay bound for each session, such as session 1 finished in time 1, session 2 finished in time 8, and session 3 finished in time 4. Although session 0 did not finished in the above 10 continuous seconds (33 pkgs left), but according to our algorithm, in the next few seconds, if the channel condition of session 0 becomes good, we can allocate the whole channel bandwidth to session 0 to keep the delay bound.

Since eventually each session can finish such as session 1 finished in time I above, that means during a busy period long enough, if a session becomes error-free, as long as it has enough service demand, it should get back all the services "lost" while it was in error (long term fairness).

As for the short term fairness, we use time 0 to illustrate. In time 0, only session 0 and session 2 are in good channel condition, we first allocate 5 packets to session 0 and 1 packet to session 2 to keep the delay bound, since the entire channel capacity are 27 packets, so we left 27-5-1=21pkgs to allocate. According to the delay bound, we allocate $5+\lfloor 21*5/6 \rfloor$=22pkgs to session 0 and $1+\lceil 21*1/6 \rceil$=5pkgs to session 2. This matches what we have proofed before:

$$\alpha * i - \beta * j \leq (E - di - dj) * ((di - dj) / (di + dj)) \quad (10)$$

at time *t*.

Table 2. The Assumption of 10 Continuous Seconds for The Channel.

time:0	time:1	time:2	time:3	time:4
0.660471	-0.443905	-0.22056	-0.74556	-0.68306
-0.157633	0.901269	-0.6435	0.440548	0.519471
0.28673	0.368034	-0.11631	0.799823	-0.78878
-0.639291	-0.284106	-0.15745	-0.7795	0.750443
time:5	time:6	time:7	time:8	time:9
-0.52759	0.104285	-0.68214	-0.89028	-0.9523
0.400079	-0.89913	0.526064	0.478087	0.449338
-0.13542	-0.65248	-0.3648	0.678173	-0.278
-0.70009	-0.94058	-0.35699	-0.1265	-0.8116

Table 3. The Simulation Results of 10 Continuous Seconds.

S / T	Session 0				Session 1				Session 2				Session 3			
	α	**β**	**τ**	**σ**	**α**	**β**	**τ**	**σ**	**α**	**β**	**τ**	**σ**	**α**	**β**	**τ**	**σ**
Time 0:	22	62	16	17	0	18	0	-20	5	56	60	4	0	23	22	-1
Time 1:	0	62	15	12	18	0	-1	0	9	47	59	12	0	23	21	-2
Time 2:	0	62	14	7	/	/	/	/	0	47	58	11	0	23	20	-3
Time 3:	0	62	13	2	/	/	/	/	27	20	57	37	0	23	19	-4
Time 4:	0	62	12	-3	/	/	/	/	0	20	56	36	23	0	18	18
Time 5:	0	62	11	-8	/	/	/	/	0	20	55	35	/	/	/	/
Time 6:	27	35	10	14	/	/	/	/	0	20	54	34	/	/	/	/
Time 7:	0	35	9	9	/	/	/	/	0	20	53	33	/	/	/	/
Time 8:	0	35	8	4	/	/	/	/	20	0	52	52	/	/	/	/
Time 9:	0	35	7	-1	/	/	/	/	/	/	/	/	/	/	/	/
.....	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Note **1**: α: how many bandwidth assigned to this session in time *t*; β: how many packets left till time *t*; τ: how many time left to keep the delay bound for this session; σ: how many packets leading (positive) or lagging (negative) till time *t*;
Note **2**: Channel capacity: E = 27 pkgs, (Since our sessions generation are 3, 1, 2, 2, so 5 + 20 + 1 + 1 = 27 pkgs), 27 pkgs are the lower bound of the channel capacity for these 4 sessions to keep the delay bound (if the channel conditions are always good).

As for the throughput, in every time unit we attempt to give good channel condition sessions more bandwidth so as not only to keep the delay bound of that session, but also to maximize the throughput of that time unit such as time 0 (24 + 3 = 27) above unless all sessions are in bad channel condition such as time 2 above.

CONCLUSION

In this paper, we propose a packet routing algorithm which can support fairness for each user and the emergency bound for each service of multi-services in wireless sensor networks. In order to generate the maximum throughput for the single channel of each time unit, we investigate the channel condition in our model. We also consider both the long-term and short-term fairness in our algorithm, so the overall throughput can be maximized. We will consider more complicated channel conditions as research emphasis in the future.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENTS

This work was supported by Jiangsu Provincial Natural Science Foundation of China (Grant No. BK20141474). We also received two Project Funded by the Forestry SanXin Project of Jiangsu Province under Grant LYSX[2014]07 and the Research of Modern Educational Technology of Jiangsu Province of China under Grant 2014-R-29605.

REFERENCES

[1] J. Cao, and T. Pei, "One Novel Multi-Services Scheduling Algorithm in WCDMA System," Proceedings of the 27th Chinese Control Conference. Kunming,Yunnan, China: July 16-18, 2008.

[2] M. A. Elshaikh, M. Othman, S. Subramaniam, R. Johari, "Enhanced TSWTCM to Improve Fairness in DiffServ Networks," *13th IEEE International Conference on Networks, IEEE MICC-ICON Proceeding.* Kuala Lumpur, Malaysia: pp. 302-307, 2005.

[3] M. A. Elshaikh, M. Othman, S. Shamala, and J. Desa, "A New Fair Weighted Fair Queuing Scheduling Algorithm in Differentiated Services Network," *IJCSNS International Journal of Computer Science and Network Security*, vol. 6 no.11, November 2006.

[4] Y. Cao, V.O.K. Li, and Z. Cao, "Scheduling Delay-sensitive And Best-effort Traffics in Wireless Networks Communications," IEEE International Conference*,* vol.3, pp 2208 -2212, May 2003.

[5] B. G. Lee, "QoS Support by Using CDF-Based Wireless Packet Scheduling in Fading Channels," *IEEE Transactions on Communications*, vol. 54, no 5, pp. 955, 2006.

[6] L. Xin, E.K.P. Chong, and N.B. Shroff, "Optimal Opportunistic Scheduling in Wireless Networks," *Vehicular Technology Conference*, vol 3, pp.1417 -1421, 6-9 Oct 2003.

[7] J.H. Abawajy, "Job scheduling policy for high throughput computing environments," *Parallel and Distributed Systems,* pp 605-610, Proceedings. Ninth International Conference, 17-20 Dec 2002.

[8] A. Farrokh, and V. Krishnamurthy, "Opportunistic Scheduling for Streaming Users in High-speed Downlink Packet Access (HSDPA)," Global Telecommunications Conference, vol 6, pp 4043-4047, 29 Nov - 3 Dec 2004.

[9] L. Xin, E.K.P. Chong, and N.B. Shroff, "Transmission Scheduling for Efficient Wireless Resource Utilization with Minimum-performance Guarantees," *Vehicular Technology Conference,* 2001. VTC 2001 Fall. IEEE VTS 54th , vol 2, pp. 824-828, 7-11 Oct 2001.

[10] Y. Liu, S. Gruhl, and E.W. Knightly, "WCFQ: An Opportunistic Wireless Scheduler with Statistical Fairness Bounds," *Wireless Communications,* vol 2, no. 5, pp 1017-1028, Sept 2003.

[11] M. A. Elshaikh, M. Othman, S. Subramaniam , and R. Johari, "An Enhancement of TSWTCM Algorithm to Improve Fairness in DiffServ Networks: Design and Implementation," Brunei International Conference of Engineering and Technology, vol. 3, pp.89-96, 2005.

[12] D. Avidor, J. Ling, and C. Papadias, "Jointly Opportunistic Beamforming and Scheduling (JOBS) for Downlink Packet Access," Communications, IEEE International Conference*,* vol. 5, pp. 2959 - 2964, 20-24 June 2004.

[13] Y. Choi, and Y. Han, "A Channel-based Scheduling Algorithm for CDMA2000 1xEV-DO system," *Wireless Personal Multimedia Communications*, The 5th International Symposium , vol 2, pp 621-625, 27-30 Oct 2002.

[14] E. H. Choi, W. Choi, and J. Andrews, "Throughput of the 1x EV-DO System with Various Scheduling Algorithms, Spread Spectrum Techniques and Applications," *IEEE Eighth International Symposium,* pp. 359-363, 30 Aug - 2 Sep 2004.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: March 31, 2015