TSG RAN WG 1#7bis **TSGR1#7bis(99)f11**
Kyongju, Korea
October 3-4, 1999

Agenda item: **5**
Source: **Golden Bridge Technology**
Title: **CPCH Simulations to support use of idle-aich**

Document for: **Discussion**

---

1. INTRODUCTION

This document is an extension and clean-up of R199-F46 which has been re-submitted from the previous WG1 meeting. **Note that a complete new section (8) with new results and cases have been added to this contribution**.

2. DISCUSSION

GBT has performed extensive simulations of the CPCH using the OPNET Modeler tool. We can provide the source code for other parties independent evaluation. The simulation is a protocol simulation and as such it is a system level CPCH simulation which captures the various impacts such as the traffic model, wireless link quality, mobile distribution and various CPCH channel selection algorithms on the throughput delay performance of the CPCH as proposed by various parties. GBT had presented some analytical results in WG1 in March and these simulations validate those results and extend it further.

The main motivation to present the simulation is to validate the use of idle-random method for the CPCH channel selection algorithm. The other motivation is to validate the need for TFCI in CPCH frames.

In these simulations, we justify the need to have idle-aich to approach perfect monitoring of the CPCH utilization. The delay results presented in these simulations are not conclusive and have comparative value. There are several layers involved in this simulation, i.e, Phy, MAC and SAR/RLC. There are several tunable parameters per layer which impact the result significantly. The results are also sensitive to traffic model parameters. For quick reference, the results in section 8.3 (case B) show the best delay performance obtained by some preliminary optimization efforts.

3. Simulation Assumptions

1. Results of Link Level Simulations with ITU channel model is used.
2. The preamble detection probability as a function of SNR (Table used).
3. Window-based and timer-based ARQ is used. So we have captured end-to-end delays.
4. 50-200 mobiles are randomly distributed in the coverage area of one cell.
5. The access Preamble ramp-up and the collision resolution steps are simulated.
6. Each packet is processed serially and independantly of others, i.e. aggregation of packets in the UE is not simulated.
7. The following tunable parameters exist in the simulations:
   - N_Max_Frames: maximum length in frames of individual packet
   - Number of ramp-ups max: number of AP power ramp up cycles without APCH response before access is aborted and packet transmission fails.
   - Traffic model: includes packet inter-arrival time, session inter-arrival time, # of packets per packet call, number of packet calls per session, Session length, average packet size, etc.
   - Three various CPCH channel selection algorithms
8. The following traffic model is used in the simulations:
   - Average packet size: E-mail application 160, 480, 1000 bytes
   - # of packets in a packet call = 15
   - Packet call inter-arrival time = 0,120
   - # of packet calls within a session =1
   - Average inter-packet arrival time = 10, 30, 100, 200 ms
   - CPCH channel data rates: 2.048 Msps (512 kbps), 384 ksps (96 kbps), 144 ksps (36 kbps), 64 ksps (16 kbps)
   - Session arrival = Poisson
9. The following results are captured:
   - End-to-End Delay, D(e-e), includes UL retransmissions and DL ACK transmission. D(e-e) is the time between the generation of the packet to the destruction of the packet.
   - Tw = Waiting time in the queue prior to transmission
   - RLC queuing delay, QD: The amount of time the packet resides in the buffer prior to being ACKed.
   - Radio Access Delay, AD
   - MAC collisions, event count for event in which 2 UE attempt access to same CPCH channel in same slot
   - Throughput (S1) includes ARQ re-transmissions/ excludes detected MAC collisions/excludes undetected collisions as well
   - Throughput (S2) excludes ARQ re-transmissions / excludes MAC collisions
   - Offered Load (rho), total offered traffic normalized to total available capacity (bandwith)
   - Undetected collisions per sec.
   - Detected collisions per sec.

4. CPCH Channel Selection Algorithms

The three CPCH channel selection algorithms are: Simple, recency, idle-random

A. Simple CPCH channel selection algorithm:

In this method, the UE monitors the available capacity and the highest available rate from the Base Node. The UE then picks a CPCH channel and a slot randomly and contends for the CPCH.

B. The recency table method:

In this method, the UE monitors the AP-AICH and constructs a recency table, which includes time-stamps, which aid the selection of the CPCH channel. The simulation assumes perfect knowledge of the transmission of AP-AICH (CPCH channel transition from idle to busy) from the base Node. In reality, there will be discrepancies in the information in the table since the UE is required to receive FACH and DL-DPCCH (while transmitting on the UL CPCH) and thus will may not be able to receive all AP-AICHs. The UE selects the CPCH channel with the oldest AP-AICH timestamp.

C. The idle-random method:

In this method, the UE monitors the idle-AICH (channel idle) and AP-AICH (channel busy) and has perfect information on the availability of the CPCH channels. The UE monitors the AP-AICH and CD-AICH for 10 ms. then it picks a CPCH channel randomly from the available ones in the desired data rate category. Note that this method is sensitive to back-off methods. When the traffic load is high and there are multiple CPCH channels, this method outperforms the other methods given the right back-off parameters.

5. Simulation Results

Cases A-B: Comparison of idle-random method and the recency method for 30 ms packet inter-arrival time, 480 bytes, and 6 CPCH channels, each @384 ksps:

We ran over 36 cases to compare the throughput delay performance of the two methods when the packet inter-arrival time is 30 ms. This was done for various packet lengths (158 bytes, 480 bytes, 1000 bytes, 2000 bytes), various rates (6 CPCH @ 384 ksps, 16 CPCH @ 144 ksps, 32 CPCH @ 64 ksps), various N_Max_Frames (8,16,24,32,64), and the three CPCH channel selection algorithms. In all cases, the idle-random method performed better. When the packet inter-arrival time was increased, the throughput delay performance of the recency method almost overlapped with the idle-random case (see Scenarios C-D-E).

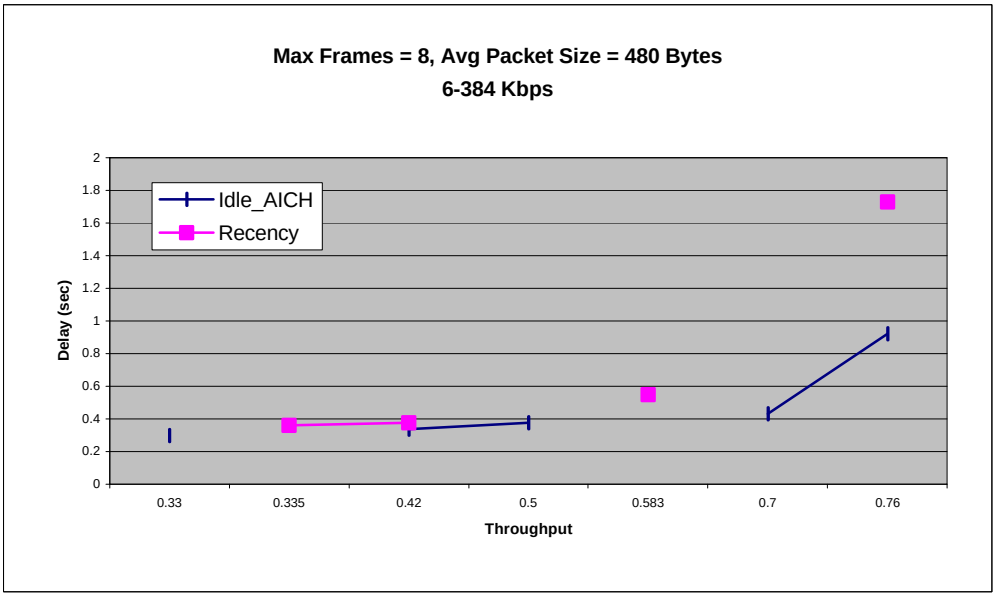
Results presented here compare idle-random method and the recency method for 30 ms packet inter-arrival time, 480 bytes, and 6 CPCH @384 ksps:

Table A: Idle-random case:

ρ	S1	**D(e-e)**
.34	.33	.3
.44	.42	.338
.53	.5	.375
.65	.70	.430
.95	.76	.92

Scenario B: recency table case:

ρ	S1	**D(e-e)**
.36	.335	.36
.45	.42	.375
.67	.583	.55
.97	.76	1.73



Scenario C-D-E: Comparison of the three methods for multiple CPCH

Recency table and the idle random methods out-perform the simple case significantly. However, the recency method performs almost as well as the idle-random case in these simulation runs for two reasons: 1) the recency table case in the simulation does not have any discrepancies in its information 2) the back-off for idle-random is not optimized and therefore it performs slightly worse when the packet inter-arrival time is high (e.g., 100 ms).

At D (un) of 400 ms, we have the following throughputs:

1. Simple case, S1 = .55
2. Recency table: S1= .8
3. Idle-random S1 = .78

Tables C-D-E (Comparison of the three CPCH channel selection algorithms)

Packet Inter-arrival time = 100 ms

1) Maximum Frame Per Packet 6

2) Avg Pk Size=480

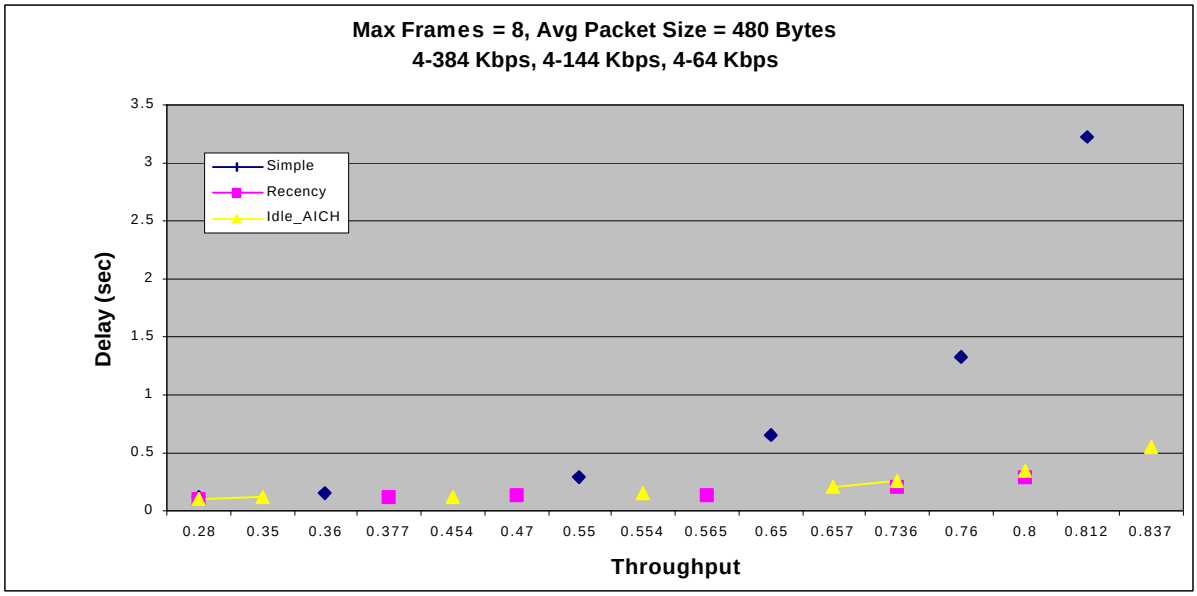
3) 12 channels:4-384CPCH , 4-144CPCH , 4-64CPCH

email_1_simple						

Sess	ρ	S1	Tw	QD	AD	TD	MAC Collision
20	0.310	0.280	0.073	0.070	0.013	0.038	677,000.000
16	0.390	0.360	0.146	0.100	0.015	0.039	106,000.000
10	0.630	0.550	.412	0.237	0.020	0.042	266,000.000
8	0.776	0.650	1.1	0.589	0.025	0.045	436,700.000
6.8	0.923	0.76	2.4	1.245	0.033	0.046	714,700.000
6.6	**1.00**	**0.812**	**7.21**	**3.15**	**0.036**	**0.047**	**983,300.000**

Email_1_recency							
20	0.283	0.280	0.077	0.062	0.009	0.038	96,500.000
16	0.380	0.377	0.09	0.069	0.010	0.038	162,000.000
12	0.477	0.470	0.112	0.081	0.012	0.038	251,000.000
10	0.566	0.565	0.124	0.088	0.014	0.038	354,700.000
8	0.779	0.736	0.242	0.149	0.016	0.038	733,300.000
7.1	**0.846**	**0.800**	**0.410**	**0.235**	**0.017**	**0.038**	**860,000.000**

Email_1_random							
20	0.282	0.280	0.066	0.056	0.007	0.039	65,100.000
16	0.351	0.350	0.1	0.072	0.007	0.039	89,000.000
12	0.458	0.454	0.104	0.076	0.008	0.040	137,500.000
10	0.558	0.554	0.169	0.109	0.008	0.041	215,000.000
8	0.667	0.657	0.269	0.160	0.009	0.042	344,000.000
7.1	0.741	0.736	0.363	0.208	0.010	0.043	472,000.000
6.5	**0.825**	**0.800**	**0.537**	**0.296**	**0.012**	**0.043**	**644,000.000**
6.3	0.876	0.837	0.920	0.488	0.013	0.043	765,300.000



Cases E-F: Impact of packet inter-arrival time

As we increase the packet inter-arrival time from 100 to 200 ms, the throughput delay performance improves significantly. As we increase the packet inter-arrival time, the packet model resembles the Poisson arrival model more. The motivation to increase the packet inter-arrival time to improve the overall delay performance of all methods. This can be achieved in practice by having the TFCI and being able to send more packets during a single CPCH transmission if it arrives in the RLC buffer. This is quite possible from a single logical channel. Note that case E corresponds to packet inter-arrival time of 100 ms presented in the previous section which is repeated here for convenience.

Table E

Idle-random, 480 bytes, 16 CPCH channels (4 @384 ksps, 4 @ 144 ksps, 4@ 64 ksps)

Packet Inter-arrival value

100

Email_1_random							
20	0.282	0.280	0.066	0.056	0.007	0.039	65,100.000
16	0.351	0.350	0.098	0.072	0.007	0.039	89,000.000
12	0.458	0.454	0.104	0.076	0.008	0.040	137,500.000
10	0.558	0.554	0.168	0.109	0.008	0.041	215,000.000
8	0.667	0.657	0.268	0.160	0.009	0.042	344,000.000
7.1	0.741	0.736	0.363	0.208	0.010	0.043	472,000.000
6.5	**0.825**	**0.800**	**0.540**	**0.296**	**0.012**	**0.043**	**644,000.000**
6.3	0.876	0.837	0.920	0.488	0.013	0.043	765,300.000

Table F

Idle-random, 480 bytes, 16 CPCH channels (4 @384 ksps, 4 @ 144 ksps, 4@ 64 ksps)

Packet Inter-arrival value

200

email_3_random							
Sess	**ρ**	**S1**	**Tw**	**QD**	**AD**	**TD**	**MAC Collision**
20	0.275	0.273	0.00	0.022	0.007	0.038	61,600
16	0.329	0.326	0.010	0.028	0.007	0.039	81,900
10	0.470	0.467	0.011	0.029	0.008	0.040	152,700
8	0.558	0.554	0.013	0.031	0.008	0.041	233,300
7	0.616	0.610	0.031	0.041	0.009	0.042	300,000
6.5	0.656	0.647	0.130	0.091	0.009	0.042	345,300
4.95	0.819	0.79	0.191	0.123	0.012	0.043	637,000
4.9	0.867	0.824	0.236	0.148	0.014	0.043	746,700

Case G: Number of mobiles in a cell:

There could potentially be hundreds of UEs in parallel session as shown by the table in this case. In third case, there are 930 UEs in parallel session if 25% of the capacity was allocated to Packet Data services. Idle-Random CPCH channel is used. There are 6 CPCH channels @ 384ksps which is equivalent to 25% of cell capacity.

Table G Delay vs Number of UEs @ 25% of cell

Packet Inter-arrival time = 200 ms

Mobiles	ρ	S1	Tw	QD	AD	TD	MAC Coll
318	.257	.256	.013	.031	.011	.038	55,766
750	.609	.604	.096	.078	.017	.042	300,000
930	.798	.772	.248	.175	.022	.044	595,000

930 mobiles in parallel session @ 25% capacity

Cases H-I: Comparison of recency and idle-random methods for single CPCH:

The recency method outperforms the random-idle for a single CPCH case and high inter-arrival time of 200 ms as shown by tables in cases F and G. The reason for this is the non-optimized back-off mechanism for the random-idle case.

Tables H-I :Comparison of recency and idle-random methods

Condition: single 2 Msps CPCH, 200 ms packet inter-arrival, 480 bytes messages

Scenario H: Idle-random case

ρ	S1	Tw	QD	AD	TD	MAC Coll
.56	.535	.282	.171	.0448	.0137	200,833
.768	.684	1.7	.883	.0729	.0137	398,000

Scenario I: recency table case

ρ	S1	Tw	QD	AD	TD	MAC Coll
.574	.634	.078	.057	.022	.0137	153,333
.813	.675	.128	.086	.031	.0136	318,666

Case H and J: Comparison of single CPCH and multiple CPCH, idle-random at 2 Msps:

As can be seen from the table the multiple CPCH case performs significantly better than the single CPCH case. Note that the packet length in the multiple CPCH case is 1000 bytes whereas in the single CPCH case it is 480 bytes. This case outperforms the single CPCH channel with the recency method as well (Case I).

Table H : idle-random case: single 2 Msps CPCH, 200 ms packet inter-arrival, 480 bytes messages

ρ	S1	Tw	QD	AD	TD	MAC Coll
.56	.535	.284	.171	.0448	.0137	200,833
.768	.684	1.7	.883	.0729	.0137	398,000

Table J: idle-random case: 4 CPCH @ 2Msps, 300 ms inter-arrival time, 1000 byte messages

ρ	S1	Tw	QD	AD	TD	MAC Coll
.57	.61	.005	.03	.012	.035	6.35 %
.76	.71	.039	.045	.016	.035	14.6%
.82	.75	.046	.05	.019	.035	18.1%
.88	.76	.174	.115	.021	.035	20%
.93	.8	.310	.184	.023	.035	23%
.975	.81	.500	.28	.025	.035	25%

6. Discussion on idle-aich and use of TFCI

As the packet inter-arrival time decreases, the throughput delay performance of all the CPCH channel selection algorithms degrades. At low packet inter-arrival times, the idle-random method clearly out-performs the recency method. The simple method performs worst in all cases. When the packet-inter-arrival time increases to 100-200 ms, then the recency method performs similar to the idle-random case. Note that at high packet inter-arrival times(very low channel loading), the throughput delay performance of all cases improves significantly. In reality, if we do not have fixed packet length and let the UE transmit the incoming packets from the higher layer midst the CPCH transmission, then the packet inter-arrival times will be higher values. By optimizing the random-idle case with appropriate back-off mechanism and incorporating the impact of the discrepancies in the recency table, the random-idle case will perform better at high packet inter-arrival

times as well. So, we propose adoption of use of idle-aich to provide for more knowledge of the CPCH channel usage.

7. Recommendations

1. Use the idle-AICH channel selection algorithm to improve the performance when the packet inter-arrival time is small.

~~2.~~2. Use of TFCI is recommended so that the packet arrival process become less clustered and approach the Poisson statistics. This will ensure better throughput delay performance.

8. Delay sensitivity simulations

We performed an optimisation study of the throughput delay performance while focusing on the idle-random channel access algorithm. The result of these studies can be summarised as follows:

1. Increase N_Max_Frame so those large packets are not segmented.
2. Increase the packet inter-arrival time by concatenating packets within a packet call thus decreasing the number of packets in a packet call. This improves the delay performance significantly. So, it is important to keep TFCI and not have an a-priori fixed packet length.
3. Higher rate DL ARQ channel decreases the end-to-end delay to some extent.
4. RLC parameters should be optimised as well. In these simulations, we studied the impact of Maximum window size on delay performance and optimised this parameter for this set of simulations.

8.1 Impact of N_Max_Frames

Idle_random method:
Average Packet Size = 960 bytes
Packet inter-arrival time = 240 ms.
Number of packets in a call = 3
Number of Ues = 160
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps

N	ρ	S1	D(e-e)	Tw	QD	AD	TD
6	.85	.79	.602	.170	.123	.03	.045
12	.84	.77	.413	.036	.0673	.0311	.066
24	.82	.755	.334	.025	.069	.033	.08
48	.82	.753	.38	.033	.076	.0345	.084

Results for N=12

ρ	S1	D(e-e)	Tw	QD	AD	TD
.391	.38	.15	-	.0275	.011	.066
.49	.47	.164	-	.0356	.0133	.066
.64	.61	.217	-	.0386	.0183	.066
.84	.77	.413	.037	.0673	.0311	.066

8.2 Impact of packet inter-arrival time

Base Case A:
Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 30 ms.
Number of packets in a call = 15
Number of UEs = 53
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps
N=6

ρ	S1	D(e-e)	Tw	QD	AD	TD
.301	.29	.5	.428	.254	.01	.035
.377	.36	.506	.42	.233	.011	-
.5	.48	.574	.624	.336	.0126	-
.623	.59	.679	.951	.5	.014	-
.7	.65	.8	.105	.55	.017	-
.822	**.75**	**1.2**	**.19**	**.995**	**.023**	**-**

Case B:
Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 120 ms.
Number of packets in a call = 15
Number of Ues = 53
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps
N=6

ρ	S1	D(e-e)	Tw	QD	AD	TD
.28	.29	.103	.045	.045	.01	.035
.35	.34	.107	.064	.055	.011	-
.46	.44	.119	.09	.068	.012	-
.68	.63	.230	.348	.2	.017	-
.75	**.68**	**.320**	**.5**	**.277**	**.020**	**-**
.82	.75	.410	.65	.355	.023	-

Case C:

Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 400 ms.
Number of packets in a call = 15
Number of UEs = 53
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps
N=6

ρ	S1	D(e-e)	Tw	QD	AD	TD
.5	.48	.085	.005	.026	.013	.035
.59	.55	.095	.013	.032	.0144	-
.67	.62	.108	.028	.041	.017	-
.77	**.7**	**.14**	**.045**	**.051**	**.021**	**-**

8.3 Impact of number of packets in packet call

Base Case A:
Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 120 ms.
Number of packets in a call = 15
Number of Ues = 53
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps
N=6

ρ	S1	D(e-e)	Tw	QD	AD	TD
.28	.29	.103	.045	.045	.01	.035
.35	.34	.107	.064	.055	.011	-
.46	.44	.119	.09	.068	.012	-
.68	.63	.230	.348	.2	.017	-
.75	.68	.320	.5	.277	.020	-
.82	**.75**	**.410**	**.65**	**.355**	**.023**	**-**

Case B:

Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 120 ms.
Number of packets in a call = 5
Number of UEs = 160
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps
N=6/12

N	ρ	S1	D(e-e)	Tw	QD	AD	TD
6	.55	.51	.14	.011	.03	.014	.035
6	.67	.622	.206	.035	.042	.015	-
6	.76	.69	.292	.070	.060	.019	-
12	.74	.69	.223	.029	.043	.022	.043
12	**.813**	**.74**	**.304**	**.131**	**.1**	**.026**	**.043**

8.4 Impact of the ARQ channel rate

Base Case A:
Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 120 ms.
Number of packets in a call = 15
Number of Ues = 53
Maximum Window Size = 4
DL ARQ channel Rate = 128 kbps (A) versus 256 kbps (B)
6 CPCH @ 384 ksps
N=6

ARQ Channel Rate	ρ	S1	D(e-e)	Tw	QD	AD	TD
128 kbps	.75	.68	.320	.5	.277	.020	.035
256 kbps	.75	.70	.288	.365	.21	.020	.035

8.5 Impact of maximum window size (MWS)

Base Case :
Idle_random method:
Average Packet Size = 480 bytes
Packet inter-arrival time = 120 ms.
Number of packets in a call = 15
Number of Ues = 53
Maximum Window Size = 1 (A), 4 (B), 7 (C)
DL ARQ channel Rate = 128 kbps
6 CPCH @ 384 ksps
N=6

Maximum Window Size	ρ	S1	D(e-e)	Tw	QD	AD	TD
1	.74	.69	27.8	42	21	.018	.035
4	.75	.68	.320	.5	.277	.020	.035
7	.816	.73	.432	.674	.366	.0231	.035

8.6 System Level Conclusions

1. Increase N_Max_Frame so those large packets are not segmented (SAR and N_Max_Frame) .
2. Increase the packet inter-arrival time by concatenating packets within a packet call thus decreasing the number of packets in a packet call. This improves the delay performance significantly. So, it is important to keep TFCI and not have an a-priori fixed packet length (SAR/TFCI).
3. Higher rate DL ARQ channel decreases the end-to-end delay to some extent (FACH or other downlink mechanisms).
4. RLC parameters should be optimised as well. In these simulations, we studied the impact of Maximum window size on delay performance and optimised this parameter for this set of simulations.
5. The results presented in case B of section 8.3 could be used as a basis for comparison with other methods. Note that the protocol can go through multi-layer optimisation and the results could be further improved.