The Supplement to "*Modeling UMTS Power Saving with Bursty Packet Data Traffic*" Submitted to *IEEE Transactions on Mobile Computing*

Shun-Ren Yang* and Sheng-Ying Yan

1 Discrete Event Simulation Model

This section describes a discrete event simulation model for UMTS DRX mechanism. According to the ETSI packet traffic model described in Section 3, a packet service session consists of N_{pc} packet calls. Each of these packet calls in turn contains a sequence of N_p packets. N_{pc} and N_p are assumed to be geometric random variables with mean μ_{pc} and μ_p, respectively. This assumption can easily be relaxed to accommodate general distributions in our simulation experiments. Our simulation model defines five types of events: **Packet arrival**, **Packet departure**, **Sleep**, **Reading**, and **Wakeup**. The output measures of the simulation are:

1. N_{sp}: the number of served packet arrivals in a simulation run
2. T_{tw}: the total waiting time of the served N_{sp} packet arrivals
3. T_o: the length of total observation period in the simulation run
4. T_s: the length of total sleep period within T_o

These output measures are used to compute the mean packet waiting time $E[t_w]$ and the power saving factor P_s:

$$E[t_w] = \frac{T_{tw}}{N_{sp}} \text{ and } P_s = \frac{T_s}{T_o}.$$

*Corresponding Author: Shun-Ren Yang, Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.; Email: sryang@cs.nthu.edu.tw

The simulation flowchart for the UMTS DRX is given in Figure 1. These events are inserted into an event list, and are deleted/processed from the event list in the non-decreasing timestamp order. A simulation clock is maintained to indicate the progress of the simulation. The clock value is the timestamp of the event being processed. In each simulation run, $N = 20,000,000$ incoming packets are simulated to ensure that the simulation results are stable.

In Figure 1, Step 1 initializes N, T_{tw}, N_{sp}, T_s and T_o to 0 and sets counters n_{pc} and n_p to the N_{pc} and N_p random variates which could be derived following Inverse Transform Technique [1] for geometric distribution. In addition, the initial server state is set to idle. The first Packet arrival event of the first packet service session is generated at Step 2 according to the inter-packet arrival time distribution and is inserted into the event list. The next event *e* is extracted from the event list at Step 3. Step 4 checks the event type of e.

Packet arrival: When a packet arrives to the RNC buffer, if $N = 20,000,000$ at Step 5, then the simulation terminates, and the performance measures $E[t_w]$ and P_s are computed at Step 6. Otherwise, the number of packet arrivals N is incremented by 1 at Step 7 and the counter n_p is decremented by 1 at Step 8. Step 9 utilizes n_p to check if the packet call in service has terminated. If $n_p > 0$, then the next Packet arrival event of the current packet call is generated and is inserted into the event list at Step 10. After that, the simulation proceeds to Step 11. On the other hand, if $n_p = 0$, it means that the last packet of the packet call has arrived, and the simulation executes Step 11 directly. Step 11 checks the server state. If the server is idle, Step 12 changes the server state to busy, and removes the Sleep event from the event list in order to stop the RNC inactivity timer. At Step 13, the corresponding Packet departure event of event *e* is generated according to the service time distribution and is inserted into the event list. Meanwhile, N_{sp} is incremented by 1. Note that, in this case, the Packet arrival event *e* does not need to wait for service, and therefore, the total waiting time T_{tw} remains unchanged. When the server state is busy at Step 11, the Packet arrival event *e* can not be served immediately and is stored in the RNC buffer at Step 14. Eventually, the simulation returns to Step 3.

Packet departure: When a packet completes the service and departs, Step 15 checks if there is

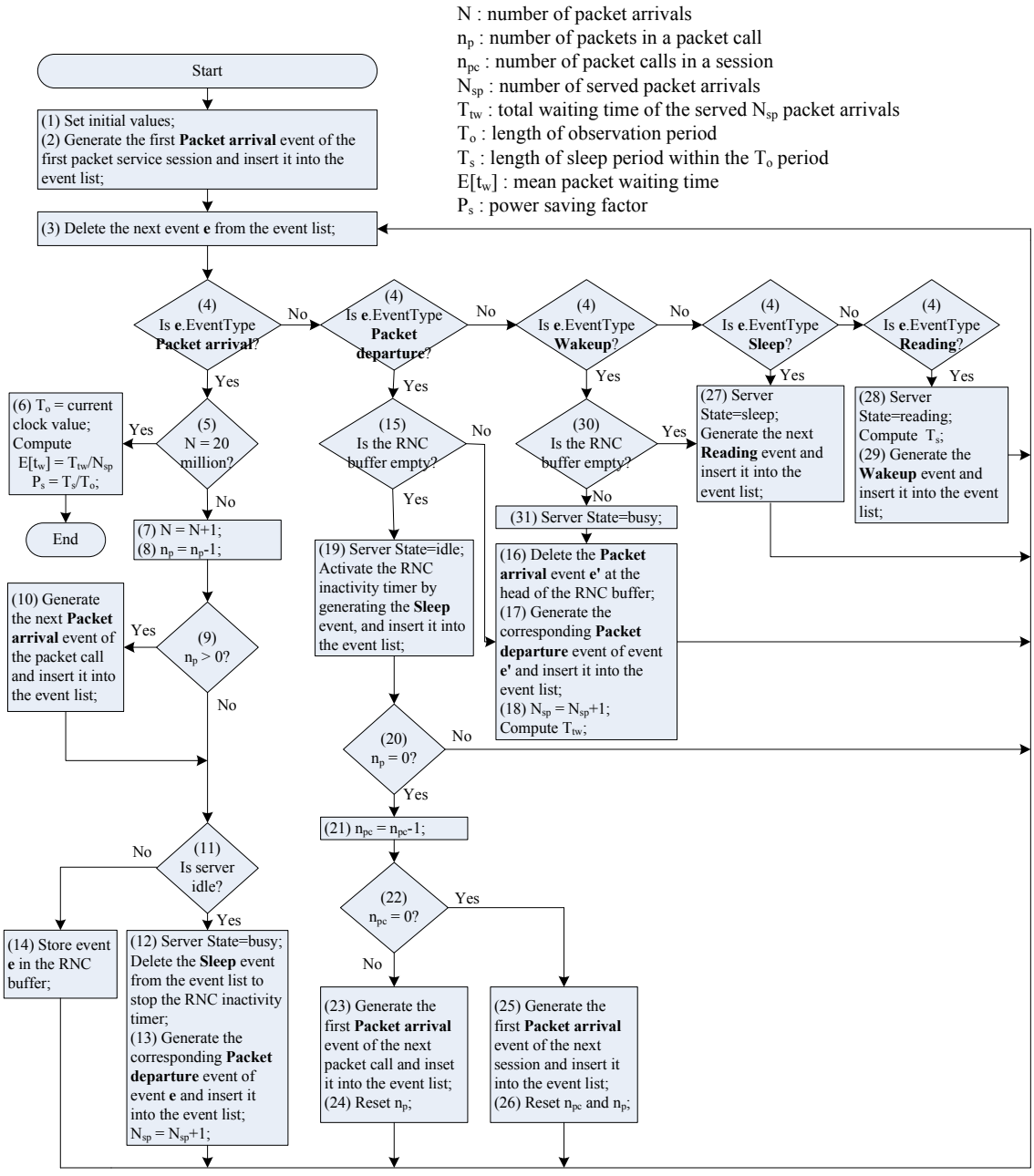

Figure 1: Simulation flowchart for UMTS DRX mechanism

any remaining packet in the RNC buffer. If the RNC buffer is not empty, Step 16 processes the Packet arrival event *e'* waiting at the head of the RNC buffer. Step 17 generates the corresponding Packet departure event for e' and inserts it into the event list. At Step 18, T_{tw} is incremented by the waiting time which event *e* spends in the RNC buffer, and N_{sp} is incremented by 1. On the other hand, if there are no packets in the RNC buffer at Step 15, the server returns to idle state. In this case, a Sleep event is generated to activate the RNC inactivity timer at Step 19, and its timestamp is calculated based on the fixed RNC inactive timer t_I. Step 20 employs n_p to check if the serving packet call has terminated. If $n_p > 0$, the simulation proceeds to Step 3 to process the remaining Packet arrival events of the current packet call. If $n_p = 0$, the current packet call has finished service, and the counter n_{pc} is decremented by 1 at Step 21. Step 22 evaluates n_{pc} to check whether the ongoing packet service session has completed. If $n_{pc} > 0$, then the session continues and the first Packet arrival event of the next packet call is generated according to the inter-packet call idle time distribution and is inserted into the event list at Step 23. Step 24 resets counter n_p to another N_p random variate for the next packet call. When $n_{pc} = 0$ at Step 22, the session has completed and the first Packet arrival event of the next packet service session is generated according to the inter-packet session idle time distribution and is inserted into the event list at Step 25. Step 26 resets n_{pc} and n_p for the next packet service session. Finally, the simulation goes back to Step 3.

Sleep: When the RNC inactivity timer is expired, the MS enters the power saving mode to reduce power consumption, and the server changes to sleep state. In order to periodically wake up to listen to the information from the BS for MS, the Reading event is generated according to a fixed sleep period t_S and is inserted into the event list at Step 27. After that, the simulation proceeds to Step 3.

Reading: The MS awakes from the sleep mode and the server state is reset reading. In addition, the length of sleep period within the T_o period T_s is incremented by the time period that the MS experiences in the sleep state at Step 28. At the end of the listening period, the MS has to determine whether going back to sleep mode or entering to active mode. Consequently,

Step 29 generates the Wakeup event according to a fixed reading period τ and inserts into the event list. Finally, the simulation returns to Step 3.

Wakeup: The MS utilizes the received traffic indication from BS to check the status of the RNC buffer at Step 30. If there are no packets in the RNC buffer at Step 30, the simulation proceeds to Step 27. Otherwise, the server changes to busy state at Step 31, and Steps 16-18 are executed as described in Packet departure case.

2 $E[W'_{1,3|2}]$ for Given t_Λ

$$\begin{aligned}
E[W'_{1,3|2}] &= \left(\frac{1}{\lambda_D} - t_\Lambda\right) + (E[N'_{1,3|2}] - 1)\frac{1}{2}\left(\frac{1}{\lambda_D} - t_\Lambda\right) + E\left[\sum_{k=1}^{N'_{1,3|2}} (k-1)\frac{1}{\lambda_x}\right] \\
&= \left(\frac{1}{\lambda_D} - t_\Lambda\right) + (E[N'_{1,3|2}] - 1)\frac{1}{2}\left(\frac{1}{\lambda_D} - t_\Lambda\right) + \left(\frac{1}{\lambda_x}\right) E\left[\frac{N'_{1,3|2}(N'_{1,3|2} - 1)}{2}\right] \\
&= \left(\frac{1}{\lambda_D} - t_\Lambda\right) + (E[N'_{1,3|2}] - 1)\frac{1}{2}\left(\frac{1}{\lambda_D} - t_\Lambda\right) + \left(\frac{1}{2\lambda_x}\right)(E[N'^2_{1,3|2}] - E[N'_{1,3|2}]),
\end{aligned}$$

where $E[N'_{1,3|2}]$ and $E[N'^2_{1,3|2}]$ are given in (27).

3 $E[W''_{1,3|2}]$ for Given t_Λ

$$\begin{aligned}
E[W''_{1,3|2}] &= E\left[\sum_{k=1}^{N''_{1,3|2}} \left\{\frac{N'_{1,3|2}}{\lambda_x} + \frac{k-1}{\lambda_x} - \frac{k}{\lambda_{ip}}\right\}\right] \\
&= \left(\frac{1}{\lambda_x}\right) E[N''_{1,3|2}]E[N'_{1,3|2}] - \left(\frac{1}{\lambda_{ip}}\right) E[N''_{1,3|2}] + \left(\frac{1}{\lambda_x} - \frac{1}{\lambda_{ip}}\right) E\left[\frac{N''_{1,3|2}(N''_{1,3|2} - 1)}{2}\right] \\
&= \left(\frac{1}{\lambda_x}\right) E[N''_{1,3|2}]E[N'_{1,3|2}] - \left(\frac{1}{\lambda_{ip}}\right) E[N''_{1,3|2}] + \frac{1}{2}\left(\frac{1}{\lambda_x} - \frac{1}{\lambda_{ip}}\right)(E[N''^2_{1,3|2}] - E[N''_{1,3|2}]),
\end{aligned}$$

where $E[N'_{1,3|2}]$, $E[N''_{1,3|2}]$ and $E[N''^2_{1,3|2}]$ are given in (27) and (29).

4 $E[t_\Upsilon|t_\Lambda]$ and $E[t_\Upsilon^2|t_\Lambda]$

$$
\begin{aligned}
E[t_\Upsilon|t_\Lambda] &= t_\Lambda + \left[\frac{1}{1-e^{-\frac{\lambda_{ip}}{\mu_p}(\frac{1}{\lambda_D}-t_\Lambda)}}\right]\left[\frac{\mu_p}{\lambda_{ip}} - \left(\frac{1}{\lambda_D} - t_\Lambda + \frac{\mu_p}{\lambda_{ip}}\right)e^{-\frac{\lambda_{ip}}{\mu_p}(\frac{1}{\lambda_D}-t_\Lambda)}\right] \\
E[t_\Upsilon^2|t_\Lambda] &= t_\Lambda^2 + \left[\frac{1}{1-e^{-\frac{\lambda_{ip}}{\mu_p}(\frac{1}{\lambda_D}-t_\Lambda)}}\right]\left\{2t_\Lambda\left(\frac{\mu_p}{\lambda_{ip}}\right) + 2\left(\frac{\mu_p}{\lambda_{ip}}\right)^2 - \left[2t_\Lambda\left(\frac{1}{\lambda_D} - t_\Lambda + \frac{\mu_p}{\lambda_{ip}}\right)\right.\right. \\
&\qquad \left.\left. +2\left(\frac{\mu_p}{\lambda_{ip}}\right)^2 + 2\left(\frac{\mu_p}{\lambda_{ip}}\right)\left(\frac{1}{\lambda_D} - t_\Lambda\right) + \left(\frac{1}{\lambda_D} - t_\Lambda\right)^2\right]e^{-\frac{\lambda_{ip}}{\mu_p}(\frac{1}{\lambda_D}-t_\Lambda)}\right\}
\end{aligned}
$$

References

[1] Banks, J., Carson, J.S. II., Nelson, B.L. *Discrete-Event System Simulation*. John Wiley& Sons, 1972.