A HIGH-PERFORMANCE, MODULAR DESIGN PARADIGM FOR TELETRAFFIC SIMULATION WITH CSIM

Sophia Scoggins
Computer Science and Telecommunications Program
University of Missouri-Kansas City
Kansas City, MO 64110
e-mail: scoggins@cstp.umkc.edu

Abstract:
The simulator designed by the author provides a new design paradigm for simulating large-scale telecommunications systems. It is based on C and CSIM and has the features of dynamic memory allocation, modular design, interruptability, user-interface, verification and displaying run-time statistics. At present, it can be used to simulate CDRs or traffic matrix inputs; Fixed Hierarchical Routing, or Spray routing; internal dynamic overflow controls for switch CPU congestion control; dynamic cross-connection by features for both switches and trunks.

1. Design Principles

Traditional general purpose simulation languages have several disadvantages in simulating large-scale systems, e.g., telecommunication systems and computer networks. Some the disadvantages are limited memory space, limited system variables, no flexibility in displaying and printing results, no interruptions, lack of flexibility in adding user functions, long simulation time, etc. This paper provides a new design paradigm for a teletraffic simulator, which does not encounter the above disadvantages, instead, it provides greater flexibility and functionality in one-tenth of the run-time in simulating large-scale telecommunication systems. This simulator is based on C and CSIM.

2. Design Paradigm

- dynamic factors vs. static factors: switch cpu's and trunk DSO ports are static factors; call, generating process, call_process, tear_down_process, user_interface and timer_process are dynamic factors.
- user_interface_process: is the initiation and front-end process, allows the user to edit, display, print, start, suspend, resume and quit simulation at breakpoints.
- call generating_processes: generating traffic.
- call_processes: represents calls.
- tear_down_processes: generated by call_processes to tear down the line simultaneously, acting as CCS packets.
- timer_process: has highest priority and records statistics at the end of each interval and each run.
- verification: verifies attributes for consistency.
- dynamic cross-connection: feature hierarchy for both homing switches and trunks.
- load shedding - switch cpu congestion controls.
- statistics analysis: real time display various statistics.

3. Performance Analysis

To process two busy hours, 240,000 CDRs in batch, the simulator took 480 seconds of cpu time on a DEC 3100 VMS workstation. The compression rate of the simulation time to real time is about 7%! In the interactive mode, the same amount of traffic would require one and one half days to process. The simulator requires 30 megabytes of virtual memory to store 10 traffic matrices of statistics. The size of the simulator in the interactive mode is close to 200 kilobytes. The batch process has half of the size. In order to verify the correctness of traffic generation from the traffic matrices, traffic are generated with an exponential distribution for every pair of LATAs. The mean of interarrival time is one hour divided by Erlang or CCS and multiplied by a mean call holding time (assumed 114 seconds). At the end of simulation, the total number of call attempts are compared to the Erlangs or CCS and mean call holding time. The mean of the difference is less than 0.5%. Similarly, the total number of generated CDRs is compared to the one of the original CDRs. So the correctness of the simulator is verified for both matrix demand generation and CDR demand generation.

4. Conclusion

The simulator has both the advantages of C and CSIM. It is dynamic, efficient, powerful and user-friendly. The modularity of C allows the simulator to expand its functionalities without much change to the base model and to maintain its efficiency.

Acknowledgements

This work is in part supported by the LiTel Telecommunications, Inc. at denton, Ohio.