Abstract

We began our project focusing on a real-world application that monitors Wide Area Networks composed by several devices of different kinds (core switches, access points, hosts, ...).

In this sort of applications many aspects of the devices must be analyzed separately (e.g. the core switches must be linked together to form a ring, the wireless devices must be reachable through the access points, the servers must have a certain amount of free memory on their disks, and so on). When a not-expected value is detected, some sort of signal should be sent to the people supposed to take care of it (e.g. by means of a ticketing system).

A naive implementation would define a separate process to monitor every single aspect of the network, but this approach would fail to recognize inherent dependencies among testing procedures. For instance, if an access point is not reachable the many wireless devices that are connected to it, would be not reachable as well. The best solution is the one that always detect only the problem at the root and get rid of the many “collateral effects”. Also it is desirable that in the system a wireless device is connected to at most one access point at the time.

In our work we recognized several safety and liveness properties expressed in indexed-LTL that characterize, in our opinion, a good network monitoring application. In order to prove them, we had to handle several characteristics of the system together:

Time Dependency Two events that are potentially related, are actually...
correlated if they happen close in time;

**Process Synchronization** Processes can cooperate by means of information exchange; the way they communicate (synchronous vs asynchronous, point-to-point vs globally, ...), affects the decidability of the model checking problem.

**Symmetry Reduction** The example shows how real-world systems may result from the composition of many copies of a few “process schemata”; this can be exploited in order to make feasible the verification procedure;

**Predicate Abstraction** It is used in our example to model the fact that, let’s say, some devices during the execution may, or may not, be linked to a certain access point;

**Cut-off Reduction** The Emerson and Kahlon’s cut-off theorems, under certain restrictions on the structure of the system, allow to obtain general results verifying only a finite configuration of the system itself.

The time-dependence correlation is of course a trade-off because when two potentially related events happen close in time they are not necessarily causally related. Indeed this is the best possible solution that can be achieved, and false positives can be reduced by a proper calibration of the considered time-window.

The verification procedure involved several formalisms and reductions so we analyzed their mutual compatibility and defined a verification workflow. The latter is made of several stages, that we name (1) process specification, (2) predicates specification, (3) model construction, (4) model reduction, in which the output of every stage provides a suitable input for the next one and the overall process is sound.

Our effort is now towards defining a formalism that handles this aspects together and drives the software engineer during the application of such transformations.