A General Purpose Network Simulator

About this document

The main idea behind this document is to lay out the requirements and potentially parts of the design for a simulator that allows to simulate processes of general nature on a network

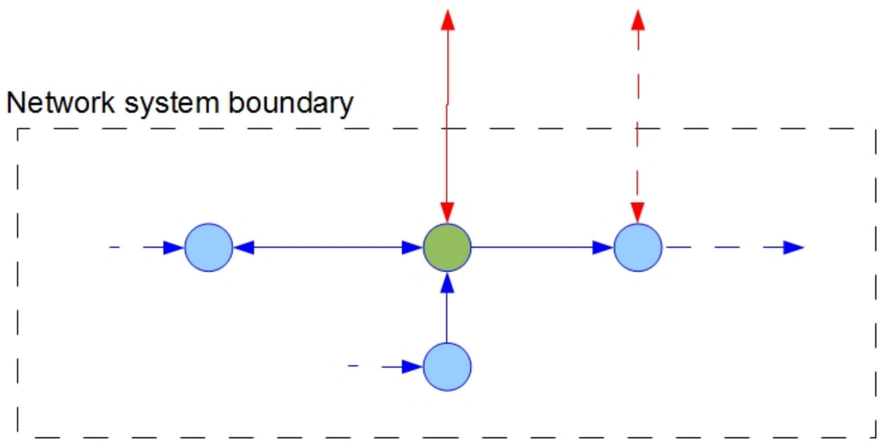
General concept

The whole network with the process(es) on it is considered the network system, in short the system. The network may be set in a virtual or real environment. In such cases, interaction is possible between the system and its environment.

Each node in a network is a subsystem on its own. It is connected to its neighbours by edges. The process(es) on the node may have inputs from and outputs to neighbouring nodes. Such are called in-network inputs/outputs. The process may also have inputs from and outputs to the environment. These are called out-of-network inputs/outputs.

Graphical representation

The following figure demonstrates the concept by focusing on a single node (in green). What is “visible” to this node is its immediate neighbours (and the state of the processes therein) and the in- and outflow from/to the node to/from the environment and the neighbouring nodes.



Detailed requirements/options

States: discrete, continuous, hybrid

The processes on the network may be of discrete, continuous or mixed (hybrid) behaviour (with respect to state vector values). In any case, at a given point in time, t, the process state in the node is characterized by a vector of values, $X_i(t)$, which can be all discrete, all continuous or mixed. In the general case, the evolution of the discrete states is simulated by Markov Chain model processes. Note that the included probabilities for transition between discrete values may depend on the full state vector, i.e., including the continuous states, state vectors of other nodes, e.g. neighbouring nodes, and external factors (via out-of-network inputs and outputs). If the process is discrete in time, write the probability for obtaining a given value, v, for the i-th state in a given node, j :

$P(X_{i,j}(t+\Delta t) = v) = f_v (X_{.,j}(t), Z_{.,1}(t),\ldots Z_{.,k}(t),\ldots Z_{.,n}(t), U_.(t))$

$Z_{.,k}(t)$: state vector in k-th neighbour (of n neigbours)

$U_.(t)$: in-network input and outputs

$V_.(t)$: out-of-network input and outputs

Of the simplest continuous-state models are state-space (SS) models (discrete in time) and Ordinary Differential Equations (DOE, continuous in time). State-space equations are of the form:

$X_{i,j}(t+\Delta t) = f(X_{.,j}(t), Z_{.,1}(t),\ldots Z_{.,k}(t),\ldots Z_{.,n}(t), U_.(t))$

Time: discrete, continuous, mixed

A special case for Markov Chain processes exist when the function f_v delivers crisp values (0s and 1s). In that case the Markov Chain process is deterministic. In that particular case, one may also simulate a time-continuous version of the Markov Chain process (otherwise intractable to integrate probabilities over time):

$P(X_i(t+dt) = v) = f_v (X_1(t) , X_2(t), X_2(t), \ldots X_n(t))$

The discrete-time version of continuous-state models (SS above) is the state-space model form:

$X_{i,j}'(t) = f(X_{.,j}(t), Z_{.,1}(t),\ldots Z_{.,k}(t),\ldots Z_{.,n}(t), U_.(t))$

Quite obviously, just like the state vector may be of hybrid form (continuous and discrete states), the time aspect may be as well. Part of the states may thus evolve at discrete times while the other part evolves continuously.

Other options

The equations above limit the continuous-time continuous-state equations to ODE equations. It may be considered to allow the more general Differential Algebraic Equations (DAE) form and/or the Partial Differential Equations (PDE) form. Note that Markov Chain simulation requires the solution to Stochastic Differential Equations. To be discussed.

Controls

For simulation purposes, control algorithms distributed in nodes can be modeled just like any of the processes above, i.e. including discrete and continuous states, either evolving over discrete time steps or continuous time steps.

Special cases

All binary

A special case exist where the process(es) in a node are discrete and have only two possible values for the state. The process is thus binary. This will allow simulate cellular automata on a network, much the same way as cellular automata on a lattice. It is noted that a completely connected graph with n nodes and equal weight edges is the same as a n-dimensional lattice of finite size (length 2 in each of n dimensions). Any lattice can be represented as a network but not vice versa. So, for some special topologies of networks, results can be extrapolated from existing research in lattices.

All discrete

A slight generalization from the binary case, is the one where the discrete states can take a finite number (>1) of values (a finite amount of possible states of the process) or an infinite number of discrete values (say a discrete stock)

All continuous

Quite logically, all processes are continuous. This is the case for a free-flow network of hydraulic tanks and pipes (1-phase flows).

Binary + other

This case may be used to represent that certain nodes are 'on' and others are 'off', i.e. where a given process is active or not. e.g. whether a node transfers information/mass/energy or not.

Special requirements

- Must allow that the network changes. A node may be deleted → process must continue somehow, mass/energy/force balances must hold. TBD.

Some goals

Goals for the simulations include:

- Evaluation of interaction between processes on networks and network topology
- Conception and evaluation of distributed control algorithms for processes on networks
- Cellular Automata on Networks (see other document)
- Conception and evaluation of distributed and/or top-down control algorithms for automated network reconfiguration. E.g., quarantaine algorithms for stopping viral processes (forest fires) and/or algorithms for promotion of viral processes (e.g. viral marketing)