# Purpose

This document contains definitions of terms used in QCNSim and its QCNSim Web interface, along with descriptions.

## Terms in QCNSim Backend

### Token

Tokens represents the active entities of the system. The dynamic behavior of the system is modeled by the movement of tokens through set of facilities. A token may represent a task in a computer system model, a packet in communication model or memory access in a memory bus subsystem model. A token may reserve (preempt) a facility or schedule activity of various durations. A token can be a single integer (customer id), an structure, (enter time, size, etc.) or another object (such as a packet).

The routing of tokens throughout the simulator is done by carefully constructing the chain of events (which event schedules with event) and by putting pertinent information within the token, i.e., the source Entity and destination Entity, or previous Entity, next Entity. Upon processing an event, a processing function can capture, for example, the appropriate facility to which request service for a token. This class, the Route class, and the Node class implement routing resources.

### Protocol Data Unit (PDU)

This class inherits from Token class and models a PDU or Protocol Data Unit. This class will extend Token and add features for computer networks, such as TTL (Time To Live) and size.

### Facility

The Facility class is used to model a resource or service center, i.e., something from which a process requests service. A facility consists of one or more servers and one priority queue, in which processes await access to the next available server. In a high-level, a facility will model a network link and possibly a BOINC server, if this BOINC server does any processing.

### Traffic Generator

The Traffic Generator class is a parent class for classes that generate traffic for the simulator. A traffic generator is typically an event generator, which generates events in time distributed according to some probability distribution. For instance, we would have exponential traffic generator, constant traffic generator, pareto traffic generator, gaussian traffic generator, random traffic generator.

Traffic generators are attached to Nodes.

Tokens or PDUs generated by the Traffic Generators will follow an explicit route, which will be attached to each Token/PDU.

### Sensor

The Sensor class inherits from Traffic Generator. This class, however, generates traffic based on seismic events. In particular, seismic events are primarily populated into an event list or event chain. When a seismic event reaches a certain Sensor, this sensor will generated a traffic event, essentially a Protocol Data Unit (or message) directed to a certain BOINC server.

As a Traffic Generator, sensors are attached to Nodes.

### Node

The Node class describes several statistics for counting PDUs and bytes received, for instance, and to measure jitter and delay for PDUs. Typically, a node measures statistics for network layer (network PDU, therefore). Entities can be "attached" to nodes: these entities are typically traffic generators (sources), links, and application servers.

This simulator works by abstracting a computer network into traffic generators, nodes, links, and possibly sinks and application servers. A traffic generator is attached to a node, which servers as the source of the PDU/token. The generated PDU/token will have a destination node, and this node will typically be the sink for the PDU. An application server may be connected to a node. This application server Entity may receive the contents of the PDU/token and process it, generating some result or more PDUs/tokens. A node with a sink simply means that node is the end destination of the PDU/token, and the PDU/token will be dropped upon receival, after proper statistics updated. Links will be attached to nodes, and each link connects two nodes. There can be many links attached to a single node. The node will perform PDU/token forwarding based on some routing rule. Forwarding consists of consulting a routing table or rule upon receival of a PDU/token, if that node is not the final destination of the PDU/token. The routing table or rule will define the next link (attached to the current node) through which the PDU/token will be forwarded. Note that this "next link" is similar to the "next hop" information on typical routing tables. "Next link," "attached application server," "traffic generators" are all Entity objects.

### Route

This class models routes within a network, to be followed by tokens or by PDUs. A route is basically a list of Entity objects a token will follow throughout the network of facilities, in a low-level point of view. In a high-level point of view, a token/PDU will follow nodes and links, each of these Entities. A typical route will indicate nodes only, however. High-level functions must abstract the nodes and populate links to complete a route.

### Link

A network link connects two nodes, and is bidirectional. A link has, as parameters, bandwidth, propagation delay, source node and destination node (these nodes can be considered interchangeable, since the link is bidirectional).

## Terms in QCNSim Web Interface

### Seismic Event Generator

This generator will produce seismic events, which in turn will integrate the event chain within the QCNSim backend.

### Client Node

A client Node, in the Web Interface, represents a Sensor attached to a Node for the backend.

Each sensor that is placed in Sam's simulator needs to create a node to which it is linked (this node will be in the same location as the sensor). Each client node must also have a route stored that it will use to send information to a destination node.

The route must be defined by the user. Traffic (or messages) generated by Client Nodes will contain this route and will be passed to the backend.

A Client Node is connected to an Intermediate Node (i.e., the backend Node within the client node is connected to the backend Node within the intermediate node).

Regions can be designated by the user to help connect several Client Nodes to one Intermediate Node.

### Intermediate Node

An Intermediate Node, in the Web Interface, is simply a Node for the backend.

Each Intermediate Node is connected to one or more Intermediate Nodes through network links. A region can be designated, in which all Client Nodes inside that region are linked to one Intermediate Node.

The connections between Intermediate Nodes and Client nodes will produce a route, which will be attached to respective Client Nodes.

### Destination node

A Destination Node, in the Web Interface, is simply a Node for the backend. They will have BOINC servers attached.

A Destination Node is connected to one or more Intermediate Nodes, or Client Nodes, via links.

## Notes

All of the work on the simulator will be focused on the Intermediate Nodes. The goal is to minimize changes to the sensors(parser makes them Client Nodes) and server (destination nodes) implementations. This means a lot of information that the Client Nodes need will be assigned by the parser. A new step of handling Intermediate Nodes will have to be added to the simulator. This step will have 3 phases:

1. Place the Intermediate Nodes. Once a node is placed you must also designate a region where all the sensors in that region are linked to it. These Intermediate Node regions can not overlap. This process will generate links between the said nodes.
2. Place the links for the Intermediate Nodes. The user will have to draw in links either between two Intermediate Nodes or an Intermediate Node and a server. The user will have to keep making links until each Intermediate Node has a path to a destination node.
3. Designate a route to a destination node. For this step the user must select what valid nodes will be used to route information from the Client Node to a Destination Node, in sequence of Client Node - Intermediate Node - Intermediate Node - ... - Destination Node. Obviously, each node pair must be connected by a direct link. (The parser will change this so instead of links the path is made of the nodes traversed.)

If any sensors are not in a region claimed by an Intermediate Node then they will send their information directly to the closest destination node.