CHAPTER

THIRTYTHREE

COLLECTING RESULTS

The following sections introduce the INET specific concepts and features for collecting simulation results. For more information in general on collecting statistics please refer to the OMNeT++ manual.

33.1 Recording Statistics

Most INET modules are already equipped with the collection of various statistics. You can find these as @*statistic* properties in the corresponding NED files. By default, many of them are already configured to be automatically recorded as either scalars, vectors, or histograms depending on the typical usefulness of the statistic. Note that it's possible to change the default recording mode from INI files as described in the OMNeT++ manual.

If the default statistics, provided by the INET modules, are not sufficient, then you can derive new modules using only NED files, and add new statistics to them based on the signals already emitted by the module. The emitted signals can also be found as @signal properties declared in the corresponding NED files.

If even declaring new statistics isn't sufficient, then you can also derive new C++ classes from the module implementations, and add new signals and/or add new statistic collection code to them. This is the most cumbersome way to collect new results, but it's also the most expressive allowing to collect any kind of statistic.

33.2 Measuring along Packet Flows

By default, INET statistics are only capable of collecting results based on the data that individual protocol modules and applications can access. Each module collects statistics independently of the rest of the network often rendering the statistics less useful in complex scenarios. For example, a TCP protocol module that communicates with multiple other TCP modules cannot distinguish between the packets based on the path they took. To overcome this issue, INET introduces the notion of packet flows.

A packet flow is a logical classification of packets, identified by its name, over the whole network and over the duration of the whole simulation. Basically, at any given moment any packet that is present anywhere in the network can be part of any number of packet flows. Packets may enter a packet flow and leave it multiple times. Different packet flows can overlap both in time and also along the network topology. Note that a packet flow doesn't necessarily have a single entry and a single exit point.

A packet flow is usually defined by active modules that classify certain packets (e.g. matching a filter) to be entering the flow, and similarly other modules that decide when packets leave the flow. Both kind of modules are inserted into the network for this specific purpose usually at the network interface level. While a packet is being part of any number of packet flows, certain INET modules (e.g. queues) are going to automatically record certain events that happen with the packets (e.g. queueing).

So far, the notion of packet flows were introduced on the packet level. That is at any given time a packet is either completely part of a packet flow or not. In fact, this is a simplification for easier understanding. This approach is clearly not sufficient in the general case, because packets can be fragmented and aggregated throughout the network, and they can traverse many different paths. Therefore the packet flow membership is actually specified on a per bit basis. The implementation takes care of efficiently representing this data, so the coherent parts of a packet that belongs to the same packet flow is marked together.

Using the packet flow mechanism, one can easily measure the timing of various things that happen to a packet (or to a part of it). The following quantities are automatically measured (if requested) along packet flows:

- total elapsed time measured from entering the packet flow
- total time spent in queues (e.g. transmission queue)
- total delay spent in various non-queue modules (e.g. interframe gap)
- total time spent in various packet processing modules (e.g. packet server)
- total transmission time of transmitters
- total propagation time spent on the transmission medium

The collected timing data is attached to coherent regions of the packets while the packets are in the packet flow. The actual measurement, that is collecting the statistical results is usually done when the packets leave the packet flow.

If the timing statistics are not sufficient, it's also possible to collect all packet events that happens to packets in the packet flow. The following packet events can be automatically collected along packet flows:

- packet is enqueued in a packet queue (e.g. transmission queue)
- packet is delayed (e.g. interframe gap)
- packet is processed (e.g. packet server)
- packet is transmitted by a wired or wireless transmitter
- packet is propagated on a wired or wireless transmission medium

The collected packet event data structure is also attached to coherent regions of the packet while the packets are in the packet flow. To actually make a measurement, a new measurement module must be implemented which processes the collected data.

In order to configure one of the above packet flow measurements, you can use the following modules:

- FlowMeasurementStarter: classifies packets to enter packet flows and starts timing measurements or packet event collection.
- FlowMeasurementRecorder: completes timing measurements, collects statistics and makes packets leave the packet flow.
- MeasurementLayer: can be added to network nodes and network interfaces for optional packet flow measurements.

33.3 Recording PCAP Traces

The easiest way to understand the behavior of a communication network on the network level is to look at the actual packets that are exchanged. INET supports recording such packet traces in the widely used PCAP and its more recent sibling the PCAPng file formats. These file formats allow analyzing the network traffic using the well-known Wireshark packet analyizer.

All network nodes and network interfaces support the recording of incoming and outgoing traffic into PCAP files via optional PcapRecorder modules. By default, this module records all packets emitted with configured signals from its parent module. For example, a recorder module put in the network interface module records all traffic specific to that network interface, and similarly a recorder put in the network node records all traffic from the given node. It's also possible to put a PCAP recorder module on the network level to produce a PCAP file that contains all network traffic. If the traffic of more than one network interface is recorded into a signle file, then the newer PCAPng file format must be used to also record the data of the corresponding network interfaces.

Recording PCAP traces also supports using packet filters, which in turn allows one to produce multiple files for the same network interface containing different kind of traffic.