Nam 1.0 Trace File Format

Mark Handley, mjh@isi.edu

1 Packet Processing

Basic packet events consist of a type, a source node, a destination node, a packet id, and a packet size:

```
<type> -t<time> -e<extent> -s<src> -d<dst> -c<conv> -i<id>
```

The values are TCL/Unix style flags as this makes the future addition of new flags much simpler. The ordering need not be this order, but it is recommended that this order is maintained to aid readability. Flags are restricted to single letters. The use of single letter flags helps reduce the size of log files, and shouldn't cause any problems. Log files are not really intended to be human-readable except for debugging purposes.

`<type>` is one of:

`h` - hop - the packet started to be transmitted on the specified link.

`r` - receive - the packet finished being received on the specified link.

`d` - drop - the packet was dropped from the specified queue or link.

`+` - enter queue - the packet entered the specified queue.

`-` - leave queue - the packet left the specified queue.

Drop here is ambiguous as to whether the packet was dropped from the link or the queue. Without qualification, "drop" means the packet was dropped from the queue feeding the link. If the packet was dropped whilst in transit on the link, this must be qualified with a -T flag as described below.

The flags have the following meanings:

`-t<time>` is the time the event occurred.

`-e<extent>` is the size (in time) of the packet.

`-s<src>` is the originating node.

`-d<dst>` is the destination node (for queue types, the queue referenced is the one at `<src>` on the edge bound for `<dst>`)

`-c<conv>` is the conversation number.

`-i<id>` is the packet id in the conversation.

`-T<droptime>` gives the time since the first bit of the packet was transmitted on the link. It is used for drop events to indicate where the packet must be placed on the link when playing backwards.

Additional flags may be added for some protocols. This list may be extended as required:

-p<ptype> gives an ascii string specifying a comma separated list of packet types. Some values are:

- TCP - a tcp data packet
- ACK - generic acknowledgement
- NACK - generic negative acknowledgment
- SRM - SRM data packet.

-n<sequence number> gives the packet sequence number or segment.

-x<payload> gives some arbrary ascii string representing the packet's payload. This can be used with nam extension scripts to be able to display the packet contents in a user-definable manner.

-#<comment> gives some arbrary ascii string as a comment as to the purpose of the packet. Can be displayed by nam to aid understanding of a simulation.

2 Link State

Links may be up or down. Other states may be added if required to represent session level concepts (such as congested).

```
l -t<time> -s<src> -e<dst> -S<state>
```

<state> gives the state transition:

-sDOWN - the link went down at this time

-sUP - the link came at this time

Note: a link going down will typically be followed by a set of packet drops for packets on that link - the drops are not assumed unless indicated in the trace file.

3 Node State

Nodes may be up or down. Their interfaces may be up or down.

```
n -t<time> -s<src> -S<state>
```

This specifies a change of state of a node.

```
n -t<time> -s<src> -d<dst> -S<state>
```

This specified a change of state of an interface.

Note: a node going down will typically be followed by a set of packet drops for packets in queues on that node - the drops are not assumed unless indicated in the trace file as being removed from the queue and dropped.

4 Route State

Routing state may need to be represented in nam and displayed.

A unicast route may be set up as follows:

```
R -t<time> -s<src> -d<dst> -D<packet-dst> -T<timeout>
```

In this context, <src> is the node that contains this route, and <dst> identifies the other end of the link. The other flags have the meaning:

-D<packet-dst> gives the final destination of packets which use this routing table entry.

-T<timeout> gives the time until this entry times out.

A multicast route may be set up as follows:

```
R -t<time> -s<src> -d<dst> -S<packet-src> -g<group> -m<mode> -T<timeout>
```

In this context, <src> is the node that contains this route, and <dst> identifies the other end of the link, even if this is an incoming interface entry. The other flags have the following meanings:

-S<packet-src> gives the original source address of data packets that match this entry. It may have a value of "*" for shared-tree entries.

-g<group> gives the multicast address for this routing table entry.

-m<mode> gives the mode - i.e, whether this is an entry corresponding to an incoming interface or an outgoing interface. It can have the values -miif or -moif.

Routing state that is removed is done so using the -x flag. This slightly modifies the meaning of the timeout value. For example:

```
R -t25.0 -s8 -d7 -g24 -moif -S* -T10.0 -X
```

indicates that an outgoing interface multicast route entry for all sources to group 24 was removed at time 25.0. At the time it was removed, the timeout value read 10.0.

5 Protocol State

We need a concept of an agent in nam to be able to associate protocol state with protocol agents rather than the node itself.

```
a -t<time> -n<agent name> -s<src>
```

This creates an agent on a node that is not associated with any interface.

```
a -t<time> -n<agent name> -s<src> -d<dst>
```

This creates an agent on a node that is associated with the specific interface to <dst>

```
a -t<time> -n<agent name> -s<src> -d<dst> -X
```

This destroys the agent.

5.1 Features

Features are essentially protocol state variables attached to agents - we use the name "feature" to distinguish between the general class and the subclass "variable".

Currently we allow three types of feature state: *timers*, *lists*, and simple *variables*. Each state variable must have a name unique within that agent.

Features may be added or modified at any time including agent creation:

```
f -t<time> -a<agentid> -T<type> -n<var name> -v<value> -o<prev value>
```

`<type>` is "l" for a list, "v" for a variable, "s" for a stopped timer, "u" for a up-counting timer, "d" for a down-counting timer.

`-n<var name>` gives the variable name. Any variable name acceptable in TCL can be used.

`-v<value>` gives the new value of the variable.

`-o<old value>` gives the previous value of the variable to allow rewind. With timer variables this gives the value just prior to this event taking place. Old value is omitted if the variable was created by with this event. New value is omitted if the variable is destroyed with this event.

Variable values are simple ascii strings obeying the TCL string quoting conventions.

List values obey the TCL list conventions.

Timer values are acsii numeric values. Timers may count up or down, and start counting immediately after they are set. Alternatively they may be stopped at a value.

Features may be deleted using:

```
f -t<time> -a<agentid> -n<var name> -o<prev value> -X
```

An example of the modification of a variable:

```
f -t10 -a1 -Td -nmycounter -v200 -o0
```

Sets the downward counting counter called `mycounter` on agent "10" to be 200. It's previous value was zero.