NRCI: Networked Resources for Collaborative Improvisation
Network protocol specification
Christopher Burns and Greg Surges
updated 02.23.2008 (NRCI release 4)

The NRCI network protocol is designed to enable the exchange of control data and communications among laptop performers participating in improvisation and other kinds of ensemble performance. Design objectives for the protocol include simplicity, flexibility, and stylistic neutrality, supporting our vision of the network as a resource for the transformation of musical data. Integrated functions include data request/exchange, autonomous status reporting, and instant messaging.

NRCI networking uses Open Sound Control (http://www.opensoundcontrol.org) as its underlying protocol; OSC is in turn built on top of UDP. By default, NRCI messages are broadcast over a local subnet (255.255.255.255) using port 9999. This address allows for fast and easy setup in various performance locations - there is no need to specify a unique address for each wireless network that an ensemble may use.

All NRCI network messages are formatted according to the following convention:
/receiver /sender /message-type message-values
where *message-values* can be one or more numeric or string values.

/receiver and */sender* are both network addresses (in the form of usernames established by implementations of the network protocol. Valid */receiver* and */sender* addresses depend on the implemented membership of an NRCI network; address strings built into the NRCI Pd implementation (as of release 3) include */adam*, */bill*, */brent*, */chris*, */dale*, */david*, */dylan*, */glenn*, */greg*, */jerod*, */joel*, */jon*, */kevin*, and */steve*. Any valid */receiver* should also be a valid */sender* and vice versa.

/all is a special category of */receiver*; any message sent to */all* is broadcast to every valid address. By definition, chat messages and requested data reports are broadcast using the */all* address. */all* is not a valid */sender* option.

Timing is implicit in NRCI messages; all messages represent the current state of the */receiver*, and network latency is assumed to be unproblematic. (So far, this has been our actual experience in performance!)

At present three are there categories of interface - the request protocol, the command protocol, and the chat protocol.

Request protocol:
Initiating requests for data

> */receiver /sender /duration-request 1*
>
> */sender* requests a stream of duration values (formatted as floating-point numbers representing duration times in milliseconds) from */receiver*.
>
> */receiver /sender /onset-request 1*
>
> */sender* requests a stream of onset values (formatted as floating-point numbers representing inter-onset times in milliseconds) from */receiver*.

These inter-onset values are “stale”; that is, the reported inter-onset time represents the time in milliseconds between the last reported onset and the currently reported onset. While these values may have utility, we typically conceive of onset values as rhythmic triggers - “something is happening now”. In Pd terms, onset output should typically be thought of as something to be converted to a “bang” message, rather than used as a numeric value.

/receiver /sender /pitch-request 1

/sender requests a stream of pitch values (formatted as floating-point MIDI pitch values, with 60 representing middle C, 61 representing a half-step above, and fractional numbers representing microtonal values) from */receiver*. In Pd, use “ftom” to convert frequency values in Hertz into this representation, and “mtof” for the conversion from MIDI to frequency.

Implementation notes:
If */receiver* is not currently broadcasting such a stream to */all*, it should begin to do so in response to this message, noting that */sender* has an active request. If */receiver* is already broadcasting a stream, it should continue to do so, while adding */sender* to the list of active requests. On the requesting side, */sender* should begin to parse reports send to */all* corresponding with the request (the report stream may have been active previously if there were other requests for the same data).

Ending a request for data:
/receiver /sender /duration-request 0
/receiver /sender /onset-request 0
/receiver /sender /pitch-request 0

/sender cancels a previous request for a stream of report values from */receiver*.

Implementation notes:
Request cancellations should only be sent by */sender*s with an active request (i.e., the */sender* should keep track of its own requests, and only cancel active requests). The */receiver* should remove the */sender* from the list of active requests for the stream; if there are no active requests remaining, then broadcasting ceases.

Reporting requested data

/all /sender /amp-report value

Reports */sender* overall output amplitude in dB (with zero representing silence).

/all /sender /duration-report value

Reports duration values (in milliseconds) of all “notes” or “events” produced by the */sender*.

/all /sender /onset-report value

Reports inter-onset time (in milliseconds) between rhythmic events produced by the */sender*.

/all /sender /pitch-report value

Reports pitch values (formatted as floating-point MIDI pitch values) produced by the */sender*.

Implementation notes:

Stream values are broadcast as close to immediately as possible, whenever a new value is available to report.

Note that there is no matching */amp-request* message for */amp-report*. All NRCI clients generate */amp-report* messages every 100 milliseconds; this data is used to visualize the sounding amplitude of each performer ("VU meters"). NRCI network implementations should support the request of amplitude data by exposing */amp-report* messages to the client on demand; however, no actual requests or cancellations should be made over the network.

Command protocol:

/receiver /sender /command command-name value

/sender sends */receiver* a message of type *command-name* with numeric *value*. *command-name* is a natural language string, presumably descriptive of the data being broadcast in *value,* which can be any floating-point number. Specific instances (and meanings/implementations) of *command-name* and *value* are to be negotiated between */sender* and */receiver*; these can be determined in advance (as in network compositions) or negotiated on-the-fly in performance (possibly using the chat protocol to agree upon the definitions of *command-name* and *value*).

Implementation notes:

The command protocol can be used on a one-to-one or one-to-many basis; implementations should support */all* as a valid instance of */receiver* for command messages.

Chat protocol:

/all /sender /chat chat-message

chat-message strings are broadcast to all NRCI clients using this message. One-to-one messaging is not currently supported.