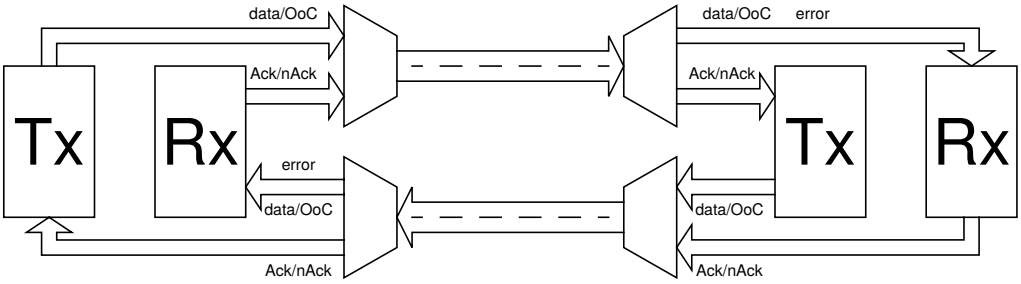
SpiNNlink frame transport



Definitions

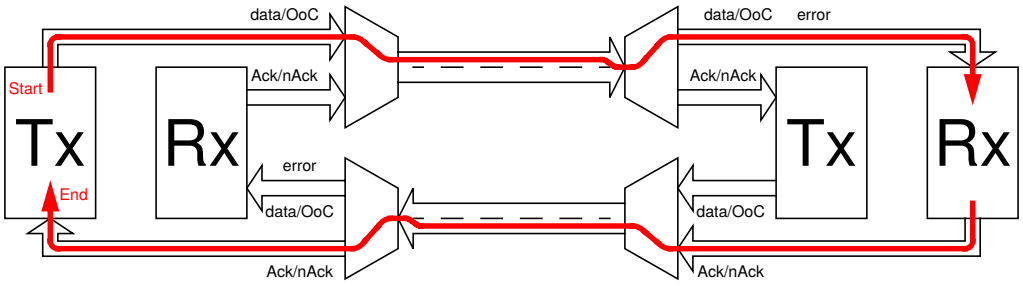
Packet	a SpiNNaker packet
Stream	a flow of SpiNNaker packets on a virtual channel
Frame	a single entity on the serial link: normally carries payload of packets etc.
Sequence	the order of frames: an increasing number
Credit	the number of frames it is known can still be sent safely
Colour	a (1-bit) ID code appended to the sequence but independent from the sequence numbers
Checkpoint	the latest state of the transmitter where the transmitted data is known to have been received

Frame 'type'	Colour	Sequence	Carries …	Function
Data	TxCol	yes	packets acknowledges/credit(?) flow control	Mixed cargo(?) payloads from higher level Frame credit return in normal operation Packet stream control
Out of Credit (OoC)	TxCol	no		Indicates that transmitter has sent everything it can and needs some form of acknowledgement
Ack	RxCol	no (? ⇒)	seq. of latest confirmed frame	Return credit to transmitter
nAck (error)	RxCol	no (? ⇒)	restart sequence No. & colour (?)	Return credit to transmitter but request resend. NOTE: sensibly needs a sync preceding
Start-up	no	no	Colour ? (Maybe system info?)	Request for 'go' signal from far end
Error	—	—	—	Never sent; result of fault on line
Sync	—	no	(may amalgamate?)	Byte-lane and packet index mark
Elastic	—	no		Opportunity for Rx FIFO to readjust

Ack and nAck may be thought of as *status reports* from the receiver for the transmitter.

In the following the synchronisation symbols {sync, elastic, start-up(??)} are ignored.

Principle of operation



- Two independent flows of data are multiplexed onto the same serial links: one is shown above.

- The data from the transmitter (Tx) is sent once, identified by a frame sequence code and protected by some form of error detection. Multiple frames can be sent successively (subject to credit limits).
  Data frames need not contain any actual data.
  If credit becomes exhausted Tx simply sends the (unsequenced) OoC frames instead.

- Received data frames are either correct or not. This status determines the state of the receiver. Erroneous frames also change the receiver colour so that subsequent (correct) frames can be flushed.
  A correct data frame will pass its error check and have the expected sequence number.
  Frames which are merely wrongly coloured may simply be ignored.

- The receiver (Rx) provides updates on its status to the transmitter at expedient intervals. These are not necessarily triggered by data arrival and continue in the absence of new data.
  Information is conveyed on the credit available, the colour and the receiver status.
  [Flow control information (Xon/Xoff or similar) for streams may also be carried.]
  Error tolerance is provided by the repetition of these ‘frames’.

- The transmitter recredits its data frames in response to the receiver status.
  (Old data, retained for possible retransmission, can be discarded up to the Ack point.)
  When an error indication (nAck) is received the transmitter changes colour – ignoring further prompts until the data stream is re-established – discards ‘old’ data up to the error point, resets its inputs to the error point and retransmits frames from the failed frame sequence point.
  There is no requirement that the data contained is the same as the previous frames; frames may be reformed with additional data if desired.

Variables

TxCol	Colour of the legitimate outgoing data frames
RxCol	Colour of acceptable incoming data frames; used for corresponding Ack/nAcks
TxErr	Flag indicating a fault is current on outgoing frames
RxErr	Flag indicating a fault is current on incoming frames
TxCredit	Number of outgoing data frames which can be sent: decremented when frame sent restored (variable amount) by Ack 'reset' to checkpoint value by nAck
RxCredit	(reflects) the space available in the (frame) Rx FIFO: decremented when frames arrive; incremented as FIFO emptied; carried by Acks

Credit

Credit is a known (may be established dynamically at start up?) number of sequence number steps beyond the last frame acknowledged. Acks contain an up-to-date seq. number and are sent at intervals, thus missing one may cause a delay but will not be fatal. An Ack can return any amount of credit, including zero.

Credit is used to throttle the rate of sending to: prevent duplication of outstanding sequence coded; to limit the data rate if the Rx FIFOs are full.

It is up to the receiver how credit is established; this could be simply the count of frames although this has limitations when credit is used for frames without data (in a lightly loaded link).

If data frames are unpacked on reception into different packet streams, it is suggested that credit be established according to the space remaining in the fullest of the stream FIFOs.

It is conceivable that more credit could be issued than data can be accommodated, speculating that capacity will become available in time. In the case of a resulting overrun it would be necessary to 'pretend' that the data frame was erroneous, thus requesting a retransmission.

Transmitter pseudocode

```
IF RxErr and <some probability>
  send nAck                 // Sent periodically whilst fault
ELSE
  IF <some probability>     // Sent periodically whilst okay
    send Ack(RxCredit)
  ELSE
    IF TxErr                // Has fault occurred?
      reset data to checkpoint
      clear TxErr
      change TxCol
    IF TxCredit > 0         // If there's credit, send something
      send data
      decrement TxCredit
    ELSE
      send OoC
```

Although Ack and Data are shown as separate alternatives here their contents are independent and they can be combined, in parallel, into the same frame.
Although – Acks must continue even if Tx OoC.

Transmitter Operation

In normal operation (error free) operation there are three possible 'frames' which can be sent. In the forward path data – with an incrementing seq. number – will be sent until the transmitter is starved of credit; data frames can be 'empty' of useful data if there is none to send.

If credit is exhausted OoC is sent until some more credit is returned or another reason becomes apparent.

Periodically – as determined locally by the transmitter – an Ack is sent to update the correspondent's credit. These are not sequenced and are sent redundantly.

If an incoming error has been detected there is an assumption that incoming data may have been lost and nAck replaced Ack. nAck is an instruction to the correspondent to rewind and repeat some outgoing frames in the original sequence but using a different colour to identify them.
nAcks are repeated until the error is cleared (by receiving the re-requested frame).
Colouring the outgoing nAcks allows the correspondent to ignore all except the first one received successfully.

If, at the construction-of-a-frame time, a request to repeat frames (nAck) has been received from the correspondent the outgoing sequence is reset to the 'nAcked' point and repeated in a changed colour.

Receiver pseudocode

```
Read frame (or other unit) from input

CASE of nAck
  IF colour = TxCol
    set TxErr
    Recredit output stream from checkpoint
  ELSE
    discard

CASE of Ack
  IF colour = TxCol
    clear TxErr             // (Needed here??)
    adjust TxCredit appropriately
    advance checkpoint
  ELSE
    discard

CASE of data
  IF colour = RxCol and seq correct
    clear RxErr
    store data
    increment RxCredit
  ELSE                      // Have missed something
    discard data
    set RxErr
    change RxCol

CASE of error
  IF !RxErr (and think this might have been a data frame)
    set RxErr
    change RxCol
  resynchronise

CASE of OoC
  IF colour = RxCol         // Missed whole new sequence
    IF RxErr
      change RxCol
    ELSE                    // Input stream starved
      ignore                // Tx periodically sends RxCredit
  ELSE
    ignore [??]
```

Receiver operation

Always try to remain synchronised with frame boundaries. If an error occurs strip further symbols until a sync (resync. mark) is detected.

On receipt of an intact data frame with the expected colour and sequence code, unpack it and move the contents to the appropriate output queues. Re-evaluate the credit state and update the value returned with Acks, as appropriate.
If the frame appears correct but has an inappropriate seq. or colour then an error has occurred.

If an error (most likely a CRC-type fault) is detected set a flag to switch to nAcks (RxErr) and discard any forwards data. Subsequent forward data will be discarded until the faulty frame is received again (in a different colour) after which normal operation is resumed.

If the retransmitted frame is also corrupted this may be picked up later by detecting a new-coloured but out of sequence frame.
If this is also missed the incoming data frames will change to repeating OoCs (in the new colour); an OoC in the error condition (assuming there is credit, and there must have been for the originally faulting frame) trigger a new attempt at nAck, changing colour again to allow one to be picked up by the correspondent. This process can repeat until the link recovers.

Responses from the correspondent (Ack/nAck) are used to adjust the available outgoing credit. In the case of nAck this also causes some outgoing frames to be repeated.

Start sequence

Following connection/full reset:

```
clear errors
zero credit
colours set to predefined initial value
Tx sends syncs/elastics to allow correspondent to lock

WHEN (own) Rx locked (& FIFO ‘centralised’)
  also start sending ‘start-up’
  (may contain system status information)

receipt of start-up indicates correspondent locked
change to normal operation:
  Tx will send OoCs with Acks passing credit to correspondent

receipt of Ack indicates correspondent has started
credit acquired: can start sending data
```

There may be less drastic reset conditions which leave error status alone, allowing starts with some nAcks and a degree of data recovery. [To be confirmed.]

Synchronisation

[Thoughts – not fully developed.]
The first state following reset is to lock the correspondent’s PLL and establish word alignment. This is done by transmitting a predefined pattern repeatedly until ‘locked’ is returned. When the start-up is complete both transmitters will assume themselves out of credit and sit correspondingly idle. (When ready) Acks will begin which will transfer credit allowing the data transmission to start.

The Tx/Rx mismatch in clock rates must be accommodated by occasionally dropping or adding symbols. Opportunity will be provided by the insertion of ‘elastic’ symbols periodically: the period being small enough to maintain lock with a maximum mismatch of clock rates plus an allowance for corrupted symbols.
It may be sensible to send these between frames, although that may not be necessary.
Receipt of an elastic symbol simply stalls the receiver process, thus discarding itself.

Syncs are used to set the phase of symbols into the output byte lanes and to indicate the start of frames. The frame format is not specified here: frames may use syncs in their headers or may rely on periodic syncs inserted into the data stream to maintain synchronisation.
Once achieved, synchronisation should be maintained unless there is a fault in receiving a frame (in particular, corruption of a data frame header resulting in a misinterpreted frame length). Following an error synchronisation should always be sought. Therefore it is sensible (necessary?) to insert at least one sync after the receipt/recognition of a nAck before the data sequence is repeated.
Periodic syncs as an extra integrity check may be sensible.

spiNNlink Frame Formats

Transmission over the high-speed links is structured in frames. The different frame formats are shown in the figure below. There are five frame types associated with data and control transmission: data (*data*), out-of-credit (*ooc*), acknowledge (*ack*), negative acknowledge (*nack*), and channel flow control (*cfc*). Each frame is identified by a different *start-of-frame* special character, highlighted in the figure, has a colour (*c)* associated with it and is protected by a CRC checksum (*CRC)*. *Frame types data*, *ooc*, *ack* and *nak* also carry a sequence number (*sequence)*. Two additional frame types, clock correction (*clkc*), and idle (*idle*), are used to keep the high-speed link synchronised.