

AM02: Records in FLEETs

Adam Megacz
megacz@cs.berkeley.edu

August 23, 2006

1 Motivation

If n words of data are to be sent from a given source to a given destination, it is desirable to configure a path through the switch fabric *once* and then transmit the words. This has three advantages:

1. The destination only needs to be transmitted once (rather than n) times, saving bandwidth.
2. The switches in the fabric need to accept configuration (which generally involves arbitration) only once, rather than n times, saving time.
3. We can guarantee that all n items arrive at the destination contiguously, even when other sources are sending datums to the same destination.

For these reasons, it is highly desirable that FLEET have a mechanism for bulk data transfers.

This memo attempts to state how records work in very precise terms. It turns out that a number of challenging problems come up when you try to “spell out” how records work, and these challenges are not always apparent at first.

2 Terminology

Word a word is the fundamental unit of data in FLEET. A Word is 72 bits long.

Record a record is a set of one¹ or more words, up to a maximum of 8 words.

3 Records and the Fabric

If we think of words as sheets of paper and the switch fabric as FedEx, then records are those nifty document mailers they give you.

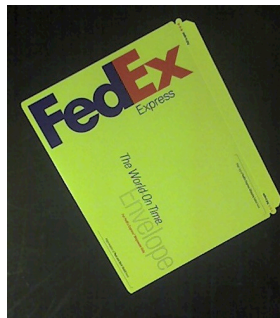


Figure 1: a FedEx mailer

Specifically:

1. Every word sent through the switch fabric *must* be part of a record, even if that record is only one word long. In practice, actual FLEET implementations are likely to optimize this case in a manner transparent to the programmer.
2. The number of words in a record is variable, and is carried within the record (ie either a length prefix or terminator suffix of some kind).
3. It is not always meaningful to talk about what happens to records after they enter a SHIP but before they leave. All that is known is that the record arrives at the SHIP, at which point the SHIP has access to the words in that record and the length of the record. What the SHIP does with such information, and how it might “re-package” the words for transmission are for the SHIP to specify, just as FedEx doesn’t care what you do with their mailers once they have been delivered (and signed for!)

¹the author reserves the right to permit zero-word records in the future

4 The MOVE instruction and records

The MOVE instruction moves a *record* from a source to a destination. FLEET will never move a “bare” word.

Records are moved contiguously – if two records are sent to a destination, each arrives intact (the words of the two records are not jumbled together, even if the records themselves arrive in an indeterminate order).

The only instruction (MOVE) now includes the following fields:

- source The source port
- dest The destination port
- count The number of *records* to move
- copy A bit; if set the record is *copied* from the source port, but remains present there. Otherwise (default behavior) the record is *drained* from the source port.
- oneword A bit; if the record waiting at the source has only one word, this bit is irrelevant. If the record has more than one word, only the first word of that record is sent (as a one-word record). If the copy bit is cleared, then the transmitted word is removed from the record at the source port.

4.1 Notation

Unless qualified, MOVE refers to an instruction with the copy and oneword bits *cleared*. The form with the copy bit set will occasionally be referred to as COPY. Each of these forms can be suffixed with “WORD” to indicate the form with the oneword bit set.

An exponent will be used to indicate a counting move (as in mathematics, a default exponent of 1 is implied when absent). As with logarithms, we will exponentiate the operator rather than the whole expression to avoid parenthesis (as in $\log^2(n + 1)$)

So, for example, the expression

$$\text{COPYWORD}^3 \quad A \rightarrow B$$

would abbreviate a MOVE instruction from *A* to *B* with the copy and oneword bits set and a count of 3.

5 Building Records

Multi-word records can be broken down into 1-word records with the oneword bit, but how do we generate long records out of short ones?

For this we will use the CONCATENATE SHIP:

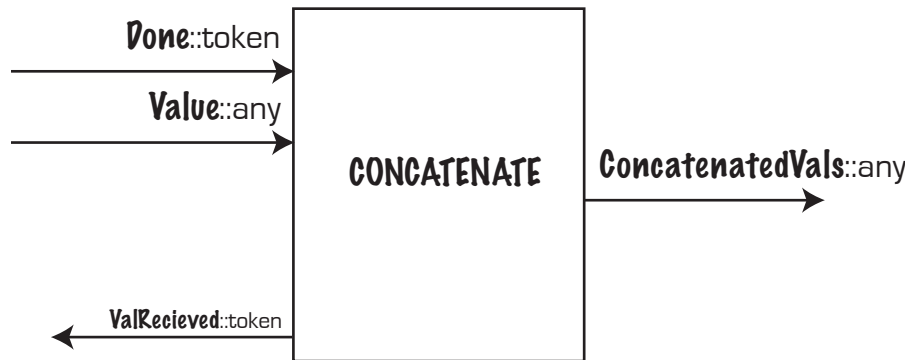


Figure 2: the CONCATENATE SHIP

Its ports are as follows:

Value Records sent here are concatenated.

ValRecieved A token is generated here after each arrival at the value input.

Done A token must be sent to this input after the last value input of a concatenation set.

ConcatenatedVals After a Done token has been recieved, the concatenated record is made available at this output.

It is crucial that the Done token be temporally ordered *after* all Value inputs. Therefore, they must be causally related. The ValRecieved output is provided for this purpose. When concatenating n records, it is the programmer's duty to ensure that the Done token is sent *as a consequence* of the emission of the n^{th} ValRecieved token.

Note that there is no need for a count input of any sort; to create a n -element record, one can issue these two instructions, which will be sequentialized due to the fact that they are in the same codebag and have the same source:

```
MOVE(n-1) ValRecieved → BitBucket
MOVE ValRecieved → Done
```

5.1 Scatter/Gather

Typically the CONCATENATE SHIP will be used in a scatter/gather configuration: if we want to concatenate records originating from sources $S_1 \dots S_n$ on SHIPs that have token inputs $T_1 \dots T_n$, we will issue the following (sequentialized) instructions for all $1 \leq i \leq n$:

MOVE WhiteHole $\rightarrow T_1$

MOVE $S_1 \rightarrow$ Value

MOVE $S_2 \rightarrow$ Value

...

MOVE $S_n \rightarrow$ Value

MOVE ValRecieved $\rightarrow T_2$

MOVE ValRecieved $\rightarrow T_3$

...

MOVE ValRecieved $\rightarrow T_n$

MOVE ValRecieved \rightarrow Done

A diagram is badly needed here.

Note that *scattering* ACKs (ValRecieveds) is the way you *gather* data (Values). This is possible because the order in which MOVEs are dispatched at a *source* is well-defined, whereas the order in which those records arrive at a *destination* is subject to the vagaries of switch fabric traffic conditions.

6 Miscillany

Every input queue on every port must be capable of holding at least one record of maximum size (8 words).