Third Idea

Dynamic Pipeline

"Sieve of Eratosthenes"

```
Master:
    for i in range (2, 1000):
        Out (("test", 2, i))
    while True:
        v = In (("prime", None))
        print v[1]

Slave (j) — initially just one thread for
                Slave (1) exists
    myPrime = j
    nextGen = True
    while True:
        s = In(("test", myPrime, None))
        if s[2] % myPrime == 0: continue
        if nextGen:
            nextGen = False
            nextPrime = s[2]
            Out (("prime", nextPrime))
            if nextPrime < 1000:
                start up new thread for Slave (nextPrime)
        else:
            Out (("test", nextPrime, s[2]))
```

This Program has several bugs — how to fix?

First bug:

Master generates

("test", 1, 2) ("test", 2, 3) ("test", 1, 4)....
("test", 1, 999)

but which will be read first?

Linda makes no guarantee about this.

Second bug:

slave (1) will effectively discard any number x such that $x \% 1 == 0$

..... but every x is divisible by 1!

We can correct the second bug by starting with slave (2) rather than slave (1)

How to correct the first bug?

- Could control sequence of tuples being read by adding a counter (FIFO order)
- Could use flow control: Master only generates new tuple when the previous one has been read.

  (same situation for other slaves)