Exercises on Distributed List Operations

Jan Plaza
SUNY Plattsburgh

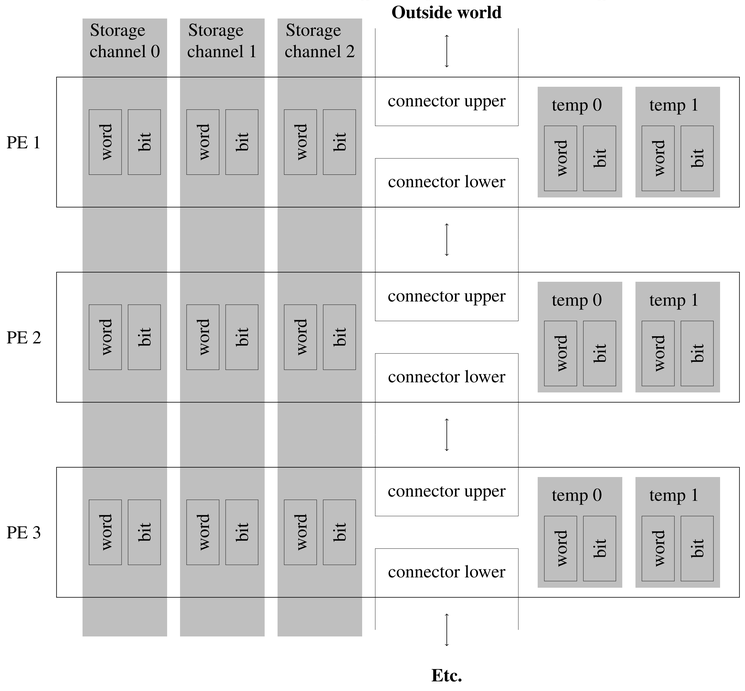
Computer Science Department
101 Broad Street
Plattsburgh, NY

jan.plaza@plattsburgh.edu
github.com/plazajan

Keywords: distributed list, distributed algorithm, line network, asynchronous chain, simulation, Python 3, complexity, domino, activity diagram, educational.

Asynchronous chain

Threads simulate asynchronous processing elements (PEs), each communicating only with its neighbors.



- A chain of PEs stores a list of words/integers, one integer per PE. Control bits tell if the word is non-empty. The first empty word terminates the list.
- The chain is infinitely extendable downwards. Only the top PE communicates with the outside world.
- Somewhat similar to a doubly linked list with access to the front, without storing the length.
- 40 exercises on distributed algorithms for list operations, from deque operations to sorting.
- Design, Python implementation, complexity analysis.
- Novel visual tools: dominos and activity diagrams.

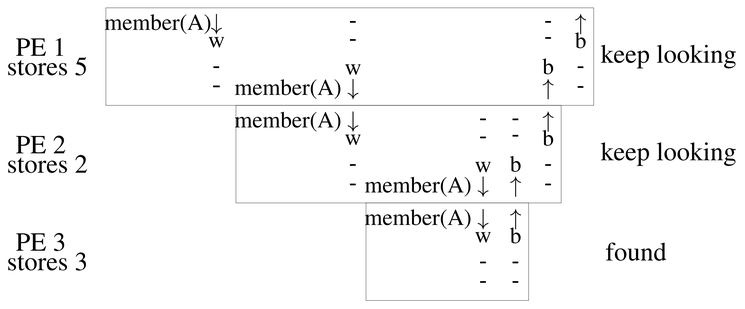
The code written by the students

A sample exercise: test if an integer received via `connectorUpper` is in the stored list. Communication commands with a suffix `_w` are for a word/integer, and `_b` for a bit. `send_o` tells the lower PE to execute the same `member` code.

```
def member(self, channel):
    if not self.bit[channel]: # if storage channel is non-empty
        self.temp_w[0] = self.connectorUpper.receive_w()
        if self.word[channel] == self.temp_w[0]: # found here
            self.connectorUpper.send_b(True)
        else: # keep looking below
            self.connectorLower.send_o("member", channel)
            self.connectorLower.send_w(self.temp_w[0])
            self.temp_w[0] = self.connectorLower.receive_b()
            self.connectorUpper.send_b(self.temp_w[0])
    else: # if current PE terminates the list
        _ = self.connectorUpper.receive_w()
        self.connectorUpper.send_b(False) # not found anywhere
```

Dominos and activity diagrams designed by students

While looking for 3 in a list [5,2,3,4,1], PEs execute a sequence of communications of types represented by the dominos in the activity diagram below. The top and the middle dominos are the same, except that the top one is stretched horizontally, because of the passage of time.

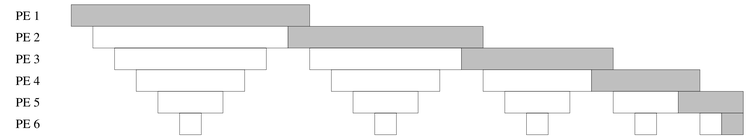


Complexity analysis by the students

For the `member` operation on a list of length n, in the worst case, the activity diagram has width $O(n)$. Conclusion: overall time complexity of `member` is $O(n)$. We introduce the concept of top complexity - the time till the availability of the top PE, which can be thought of as the width of the top domino . The top complexity of `member` is $O(n)$.

Another exercise: selection sort

Below we give an abstract view of the activity diagram of selection sort on a 5-element distributed list.



The first inverted triangle on the left, brings the smallest value to PE 1. The second triangle brings the second smallest value to PE 2, ..., the n-th triangle brings the n-th smallest value to PE n. From such diagrams, the students judge that the top complexity of the algorithm is $O(n)$, where n is the length of the list, and the overall time complexity is $O(n^2)$.

Conclusion

The asynchronous chain is suitable for practicing design of simple but non-trivial distributed algorithms aided by our visualizations, called dominos and activity diagrams, which do not adapt to arbitrary network architectures. See `github.com/plazajan` for the exercises and a detailed discussion.