Reading: Ch. 15

Next: Ch. 16, then 10 (mutual)

Last two # finished up basic asychr model ideas
Then started asychr network algo, by rewriting leader election
in ring

Today continue with (leader election, breast, coast)

\[
\begin{align*}
\text{Spanning tree} & \rightarrow \text{MST} \\
\text{in asychr networks.}
\end{align*}
\]

Recall model:

Deigraph with process at nodes

\[
\text{FIFO channel aut. on directed edges.}
\]

Last two showed LCR alg extends to asychr case.
This bond takes into account pipeline pileups (queuing delays) in channel and send buffers.

Total time still \( O(n(k+d)) \) - because pileups have simply mean that some UIDs have sped up.

(see book)

Note style of measuring time here: No ids, but using bounds on duration of time between low-level events.

*H5 - Similarly, extends to asynchronous model

\[
\text{Peterson: } \quad \frac{O(n \log n) \text{ steps}}{\text{under common}} \quad \text{hand} \quad \text{this was open G for a while} \quad (\text{H5 \& pet?})
\]

Any process, at any time, is active or relay. Relays just pass messages on.

Any phase: In each "phase", \# of active divided by 2

(at least)

So \( \leq \log n \) phases till election.

Phase 1: Read UID 2 steps + collect 2 values.

Compare: Then remain active, adopting

\( \text{the UID of the predecessor} \)

low low

(\text{the high one})

elseif inactive \( \Rightarrow \) remain inactive
Claim someone remains active.

But no more than half of those from the prior phase.

Subsequent phases:
Exactly the same, except now the inactive goes (relay) just goes on.

If ever see immediate pad's id same as yours, elect self leader.
(Know you're the only one left.)

Easy to see solves.
Easy to see # active shrinks by at least \( \frac{1}{2} \) each phase.
Not so obvious, but can show time to election is \( O(m(l+d)) \).
'Scheduled arg'; in book.

A. Love, Bd.

On comm. complexity, \( \Omega(m \log m) \), for election in async ring, follows from the sync l.b., in the special case of companion-based algo.

Actually, there's also a synchronous l.b. for general algo, but it's pretty hard (starred section).

Here, I'll give an alternative elementary proof that depends heavily on the deterministicness of asynchrony.

Illustrate very well the problems that cannot be introduced by asynchrony.
Assume ring size unknown (e.g., rounds in and size rings) VIDS, infinite space of VIDS all girls identical except for VIDS Bobin, comm.

Consider combination of girls to form rings as usual. But also lines, where nothing is connected to the ends, or no input arrives there.

Lemma 1. \( \exists \) inf set of process act, each of which can send at least 1 msg without first receiving one. (in some act)

Pf. If not, \( \exists \) 2 proc. i \& j, neither can send msg without first receiving one.

Consider must elect self; no msgs sent

Both elect control.

Lemma 2 (key). For every \( n \geq 0 \), \( \exists \) inf call \( \gamma_2 \) of function disjoint lines, s.t. for each \( L \leq \gamma_2 \),

\[
|L| = 2^n \\
c(L) \geq n^2 - 2
\]

defined to be the max (sup) \( \gamma_2 \) of msgs that can be sent in a single (input-free) act. \( \# L \)
Pf. Induction on $n$.

Base: $n = 0, n = 1, LTTR$

Inductive: show for $n \geq 2$, assuming for $n-1$.

By ind. assumption, $\exists$ inf. set $L^{n-1}$ all in $\mathbb{N}$ have

\[
\begin{align*}
|L| &= 2^{n-1} \\
C(L) &\geq (n-1) \cdot 2^{n-3}
\end{align*}
\]

Choose any $3, (L, M, N)$ for $L^{n-1}$.

Claim: at least one of the 6 combin $\{LM, ML, LN, NL, MN, NM\}$ has $C(L) \geq n \cdot 2^{n-2}$.

That's sufficient (\ldots)

Pf of claim: Suppose not, that is, can't face any of these 6 combin
to send $\geq \frac{n}{4} \log m$ (w/ $m = 2^n$)

By ind. hyp: $\exists x_L$ (finite, input-free) for $L$

\[
C(x_L) \geq (n-1) \cdot 2^{n-3} = \frac{n}{8} \log m
\]

Same for $x_M, x_N$.

Assume WLOG all end in "quiescent" (silent) states, where no further messages can be sent without further input.

(If not, extend...)

Now consider line $\underline{L} \rightarrow \underline{M}$

Construct new execution $\underline{LM}$.
Run $\mu_{L \leftrightarrow M}$ delaying mrgs output along boundary from being delivered until 2 pieces queue. Then let those mrgs be delivered, continue to run $\mu_{L \leftrightarrow M}$ to a quiescent state.

Number of mrgs: \[9 \times n \times \mu_{L \leftrightarrow M} \text{ gets } = 2 \left( \frac{n}{8} \right) \log \left( \frac{n}{2} \right) = \frac{n}{4} \log (n - 1)\]

Claim in running to quiescence, number of add'l mrgs < $\frac{n}{4}$.

(Since otherwise would total $\geq \frac{n}{4} \log n$.)

Thus, mrgs can propagate from boundary only to distance $< \frac{n}{4}$ before quiescence.

This says if the function doesn't reach the midpoint of either $L$ or $M$, before quiescence.

Same for $ML, LN, NL, ...$

Now get contradiction by considering some rings:

$L \rightarrow R$ on line corresponds to clockwise on ring

LMN:

Must select a leader, WLOG say $i_1$ of between indices of $L \leftrightarrow M$
Some $i_2$ elected here. Potentially anywhere.

Since $a_3 + i_\frac{1}{2}$ are indistinguishable to $i_3$, $i_3$ would also get elected in $x_1$, which would yield two leaders in $x_1$. Continue.

Then also $i_3$ elected in $x_2$, that is, $i_2 = i_3$

Now consider LN.

Must elect some $i_4$.

Where can it be?

If in upper half, also in $x_1$.

If in lower half, also in $x_2$. Continue.
Network Search

Fundamental tasks: Constructing spanning trees to use for communication, e.g., broadcast.

Start with simple task of setting up some (arbitrary) spanning tree rooted at a given node.

Assume:
- Undirected, connected graph.
- Processes don't know size or diameter.
- UIDs with which can identify input edges that attach to some node (don't actually need these UIDs).
- Each process should report, with parent output action, name of parent in tree.

Starting point is the szemel B-S algorithm.

Recall: Sets up a BFS spanning tree, using search messages.
- Each process chooses sender of first search as parent.
- It receives.

Can run asynchronously, still gives spanning tree, but not necessarily BFS.

Do over code, p. 496-7: {search messages, only sent in response to first search, may received tasks.

Complexity: {msgs: \(O(1E1)\)
- time: all done in \((\text{linear}) (L+1) + 1\)
But note anomaly: paths produced could be longer than those.
Because mugs travel faster on longer paths than shorter ones.

in asynchronous network.

Discuss.

Msgs first: pigtailed on search msg.
Child pointers: easy to augment with responses to search mugs.
determine children.

Using precomputed tree w/ child pointers for repeated boosts & counts:
Now note the anomaly comes into play.
Can take $O(k(l+d))$ to boost, where $k = \text{height of the constructed tree}$.
Can be bigger than linear.
Worst case $n$.

Asynchronous Boost & Ack.

Can also construct the sp tree while using it to boost &
collect acks.

(to tell when the boost is done, or to coa$t an answer.)

No new code. p. 499-500

Repeat, let's leaves initiate coa$t.

Complexity: mugs: $O(|E|)$
time: $O(m(l+d))$, because of anomaly of coa$t.

Breadth-first spanning trees

$G = (V,E)$, common under $L_0$

Prices don't grow in a dream

VIDS as above.
Each to repeat parent.
In such setting, synchronous BFS worked to give BF spanning tree.
But when run asynchronously (as above), just produces some sp tree, not rec. BF.

Can modify so processes correct erroneous info when hear better data, correct it. In this case, must inform other nodes about the change, so they can also correct. (relaxation alg.)

Steiner code, p. 502

Eventually stabilizes to BF spanning tree.
But no one knows when it's done, so can't produce parent outputs.

Can augment with asks for all search nodes, converge cast back to source i, to say done everywhere.

But this is tricky. Tree grows & shrinks, some process may need to participate many times. (bookkeeping needed)

But when asks finally all converge cast back to source i, really done.
Then i can broadcast signal to tell everyone else they're done.

Layered BFS

All this asynchrony leads to lots of corrections, which leads to extra communication.

An idea: Slow down some of the computation, trying to keep the levels of the tree more synchronized.

This in turn reduces # needed corrections, which in turn reduces communication.
* Elect leader in unknown network: (with no sp. tree)
  all initiate async-heart-ach. determine max, max
  elects self leader
  with a sp. tree.
* Use arb. sp. tree to elect leader: convergent, as in
  synch setting.
For example, could develop alternative BFS alg. that proceeds in
synchronized layers
(unknown median parent announcements)

Each layer 1 sets up all the nodes at depth k in the tree.
Synchronize in between setting up each pair of layers, by

with i_o.

Phase 1: i_o sends search msgs to nodes in

Rec mark dist. as i_o sends acks
to get acks.

Phase k+1: Assume phases 1,..., k constructed. Each node at
depth k+1 knows parent i_o also those at depth
1,..., k-1 know children

Then i_o casts a neopace msg along the tree edges, to depth
k processes (current leaves).

Each of these sends out search msgs to all except their

When non-i_o process receives list search msg, makes tender its
parent i_o and sends positive ack.

Sends neg acks to all subsequent search msgs.

When depth k+1 process has acks for all its search msgs,

calls the positive acks its children.

Then broadcast request converges back up to i_o, to say they're
done.

(a also a lot saying if anything new was found).

Terminates when nothing new is found.

Produces BFS tree.


technique with formulation steps

Complexity: Simplified

Neglect local comp. time &
Assume every msg is channel delivered in the 20
Labeled: \( m_{lcp} \leq \mathcal{O}(|E| + n \cdot \text{diam}) \)

every edge explored true edges can be explored
at most twice once repeatedly, once per phase in each direction \((\leq \text{diam phases})\)

\[ \text{time: } O(\text{diam}^2 d) \]
because of diam layers

Compare with asynchronous BFS:

\[ m_{lcp}: \text{Much worse.} \]

\[ O(m \cdot |E|) \]

\( \uparrow \)

\[ \# m_{lcp} \text{ sent as a result of} \]

\[ \# \text{edges revisited} \]

\[ \# \text{edges revisited estimate} \]

\[ \text{why: } n \text{ is longest path distance estimate anyone could get} \]

\[ \text{time: better, } O(\text{diam} \cdot d) \]

no back & forth

In both, you will also see a hybrid alg. with intermediate performance.
Several layers constructed in each phase.
Within each phase, construct all layers asynchronously.

Shortest Paths

Same assumptions as BFS, plus locally known edge weights.
Now want to output shortest distance & parent.

Synch case: recall Bellman-Ford

Relaxation, corrects estimates
Can run asynchronously
Used in AsynNet in 70s

Can augment with cost as for B = 5 for termination.
But complexity is very bad (in worst case)

![Graph with nodes and edges]

Incr. decreasing powers of 2
See possible diet estimates if can take on any exactly the numbers

\[2^{k-1}, 2^{k-2}, \ldots, 0\]

Moreover, can force \( k \) to take them on in sequence!
How?

Upper paths fastest: \( 2^{k-1} \)

Then next lower may arrive at \( i_k \) reduces \( k \)’s estimate.

\[2^{k-2}\]

Then next lower may \( i_{k-2} \rightarrow i_{k-1} \) arrives, \( k \rightarrow k-1 \) reduces estimate by 2

(resets search) arrives at \( i_{k-1} \) first on upper link

\[2^{k-3}\]

Etc. Count down in binary.

Suppose all this happens fast - then can get queue of \( 2^k \) msgs

in \( i_k \rightarrow \rightarrow \rightarrow i_{k+1} \)

Exponentially may msgs, so (counting congestion) can take

exp. time to arrive at \( i_{k+1} \) (cap. in \( m \))

Mind: Unrestricted asynchrony can mess things up.