Distributed algorithms

Keith Briggs

Keith.Briggs@bt.com

more.btexact.com/people/briggsk2/

2002 October 24 1415

TYPESET IN LATEX2E ON A LINUX SYSTEM

- Asynchronous distributed algorithms (ADAs)
- Simulation techniques
- Some examples

- Asynchronous distributed algorithms (ADAs)
- Simulation techniques
- Some examples

Theme: atheism

- Network of identical nodes, with message q

Asynchronous distributed model

- Network of identical nodes, with message q
- Each knows only its neighbours

Asynchronous distributed model

- Network of identical nodes, with message q
- Each knows only its neighbours
- Each performs the same subalgorithm

Asynchronous distributed model

- Network of identical nodes, with message q
- Each knows only its neighbours
- Each performs the same subalgorithm
- Each runs asynchronously wrt neighbours

- Network of identical nodes, with message q
- Each knows only its neighbours
- Each performs the same subalgorithm
- Each runs asynchronously wrt neighbours
- $\exists$ a finite set of pre-specified messages

Asynchronous distributed model

- Network of identical nodes, with message q
- Each knows only its neighbours
- Each performs the same subalgorithm
- Each runs asynchronously wrt neighbours
- $\exists$ a finite set of pre-specified messages
- Indefinite delay before reply to message

Required: to perform some useful global actions:

Required: to perform some useful global actions:

- Reboot system

Required: to perform some useful global actions:

- Reboot system
- Detect node failures

Required: to perform some useful global actions:

- Reboot system
- Detect node failures
- Count total number of nodes

Required: to perform some useful global actions:

- Reboot system
- Detect node failures
- Count total number of nodes
- Name nodes and elect leader

Required: to perform some useful global actions:

- Reboot system
- Detect node failures
- Count total number of nodes
- Name nodes and elect leader
- Build spanning trees

Required: to perform some useful global actions:

- Reboot system
- Detect node failures
- Count total number of nodes
- Name nodes and elect leader
- Build spanning trees
- Find shortest paths

Required: to perform some useful global actions:

- Reboot system
- Detect node failures
- Count total number of nodes
- Name nodes and elect leader
- Build spanning trees
- Find shortest paths
- Compute and optimize network flows

In decreasing order of weight:

- unix processes
- kernel threads
- threads in python, java etc.
- other tricks

```
import threading

class Node:

   def init(links):
     message_queue=[]
     # ...

   def run():
     while 1:
       # ...

   def send(target):
     # ...

   def receive(source):
     # ...

nodes=[Node([2,3]),Node([3]),Node([1])]

for node in nodes:
  Thread(node.run).start()
```

All nodes *asleep* except 0, who is *awake* and sends to all neighbours

- if receiver awake: return ‘reject’
- if receiver asleep:
  - wake up and relay message to neighbours
  - return number of nodes from relay replies
  - receiver returns sum$+1$ to requester

Asynchronous Bellman-Ford algorithm:

$$x(i) \leftarrow \min_{j \in \text{neighbours of node } i} x(j) + d(j)$$

where:

- $x(i)$ is node i's current estimate of the shortest path to node 0
- $d(j)$ is the distance to node j (one hop)

Termination?

Root node has weight 1

`while 1:`

- node sends its weight to neighbours
- if receiver is unweighted, adopt sender's weight+1
- else if receiver's weight $>$ sender's weight$+1$
  - receiver adopts new parent

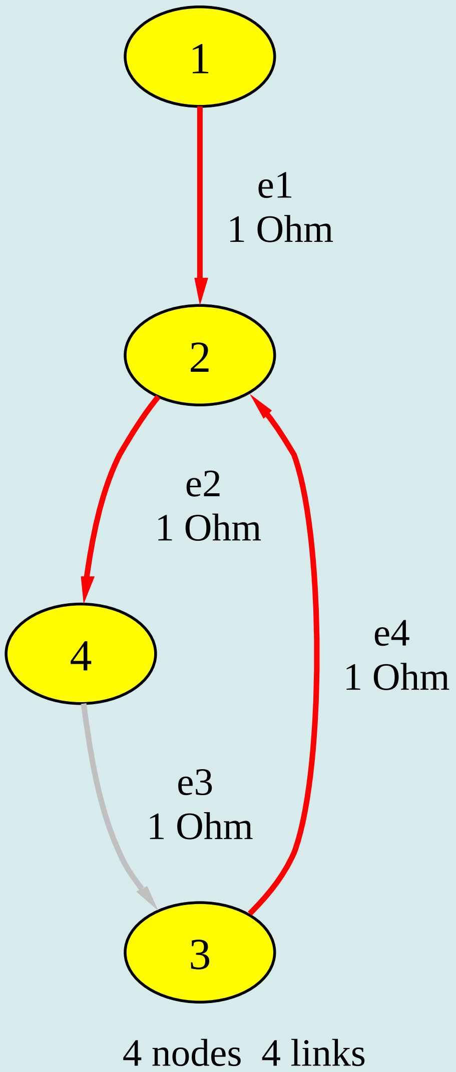
sorting, flows, . . .
routing
counting; spanning tree
reboot; failure detection
adjacency

- Digraph G with r_k the resistance of edge k
- Problem 1: translate graph topology (known only locally) to circuit equations
- Problem 2: solve these equations
- Apply Kirchhoff's current law (KCL), Kirchhoff's voltage law (KVL), and Ohm's Law (ΩL) to circuit
- Let v be the voltage vector and i the current vector (in edge space)

- A is the adjacency matrix and D is the degree matrix
- Find *incidence matrix* B from $BB^T = D - A$
  Then KCL is $Bi = 0$
- Build a spanning tree T. Edges in T are *branches*, other edges are *chords*. Each chord has a *fundamental cycle* (FC)
- C: matrix with one column for each edge, with elements being the coefficients of the corresponding FC in the edge space
- Then KVL is $C^T v = 0$
- ΩL is $v_k = i_k r_k$

- $i = Yv$, where Y is the *conductance matrix*
- $Y = -C\,C^{+R}$, $R = \text{diag}\,(r_1, r_2, \ldots)$
- C^{+R} is the *weighted Moore-Penrose pseudoinverse* of C with weight R. If $R = W^T W$, then $C^{+R} = (WC)^+ W^{T^{-1}}$
- I have developed an algorithm for incremental computation of C^{+R}, which can be applied as the columns of C are found by remote nodes

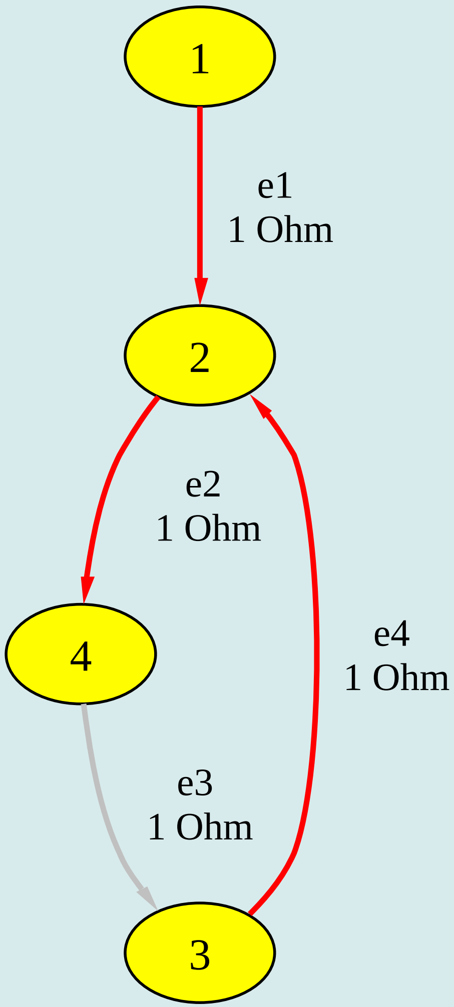
Example



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix}$$

Example cotd



4 nodes 4 links

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -1/3 & -1/3 & -1/3 \\ 0 & -1/3 & -1/3 & -1/3 \\ 0 & -1/3 & -1/3 & -1/3 \end{bmatrix}$$

- N Lynch *Distributed algorithms*, Morgan Kauffman 1996
- D P Bertsekas & J N Tsitsiklis *Parallel and distributed computation*, Athena Scientific 1997
- B Bollobás *Modern graph theory*, Springer 1998

`tantalum:/home/kbriggs/ian/Talk/distrib-algs-teratalk.tex`