A Distributed k-core Decomposition Algorithm for Dynamic Graphs

Paul Jakma[1], Marcin Orczyk, Colin Perkins[1], Marwan Fayed[2]

[1]University of Glasgow,

[2]University of Stirling

2012-12-10

A Distributed k-core Decomposition Algorithm for Dynamic Graphs

A Distributed k-core Decomposition Algorithm for Dynamic Graphs

Paul Jakma[1], Marcin Orczyk, Colin Perkins[1], Marwan Fayed[2]

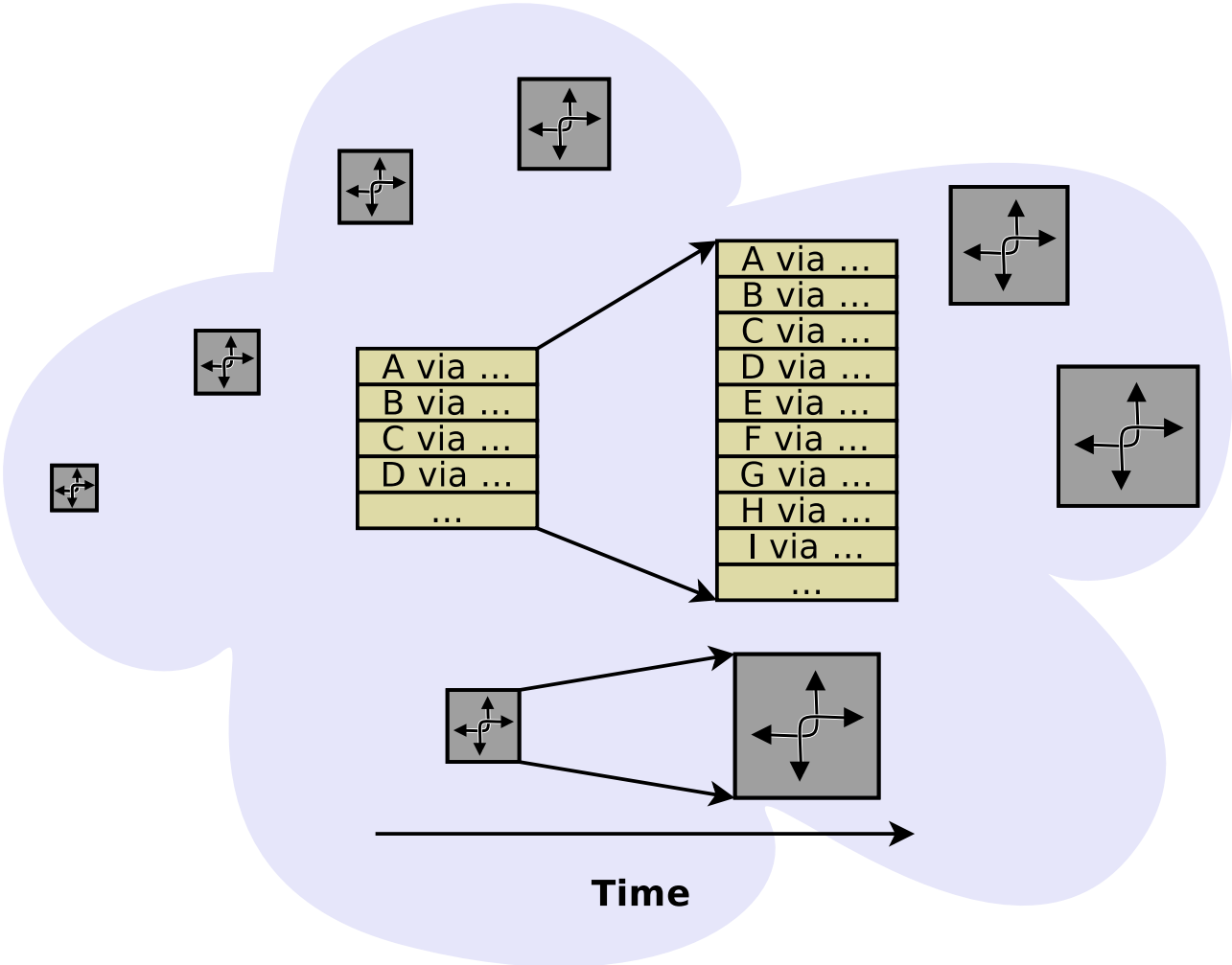
[1]University of Glasgow,

[2]University of Stirling

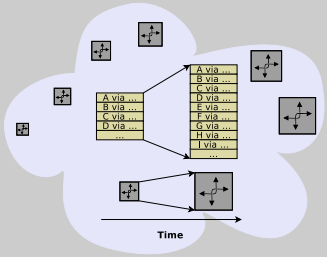
2012-12-10

Hi, my name is Paul Jakma. I'm with the School of Computing Science at the University of Glasgow, and I'm presenting a poster on A Distributed k-core decomposition algorithm for dynamic graphs. I'm going to primarily cover my motivation for this work, and then introduce our contribution.

Routing Table Scalability

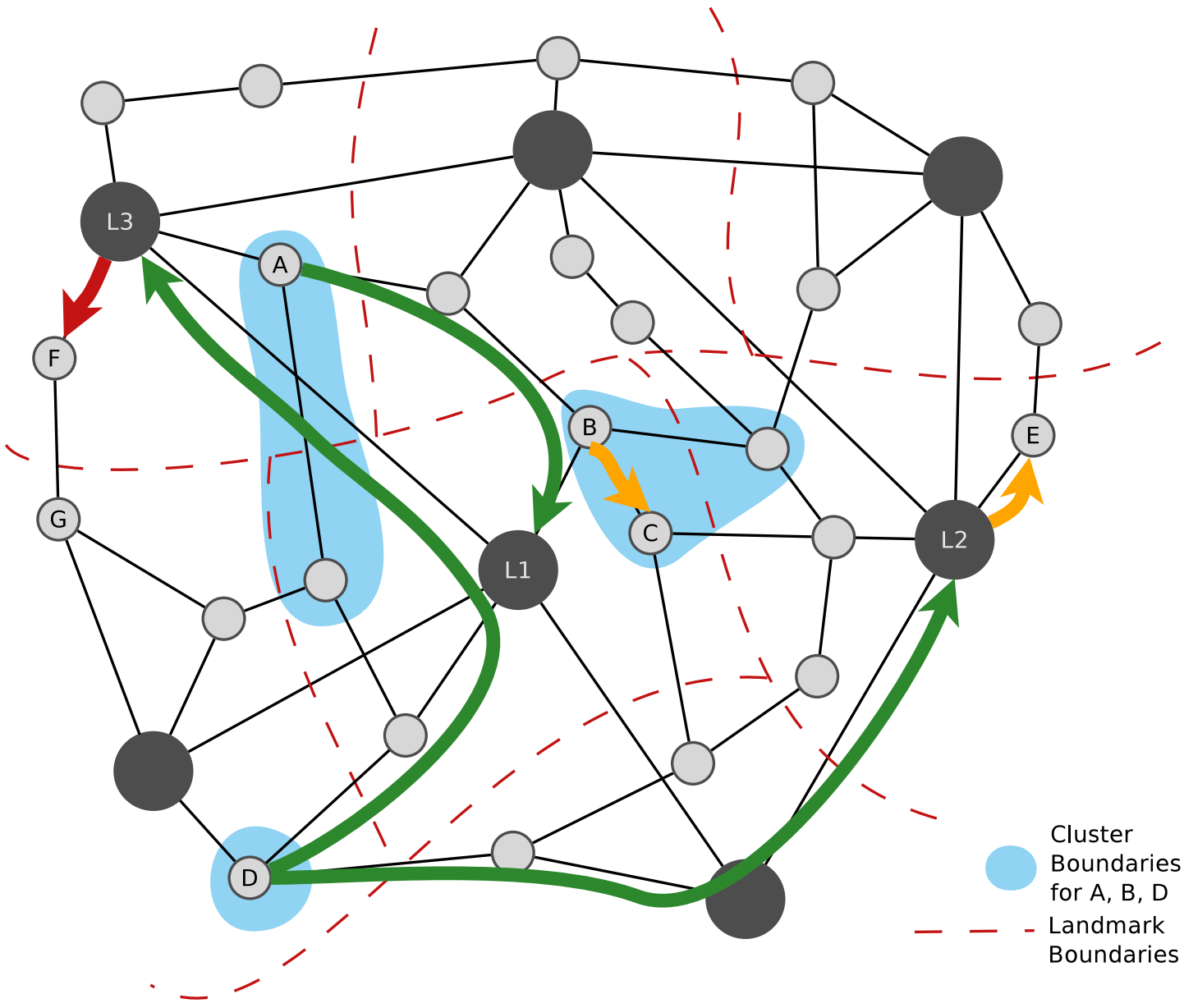


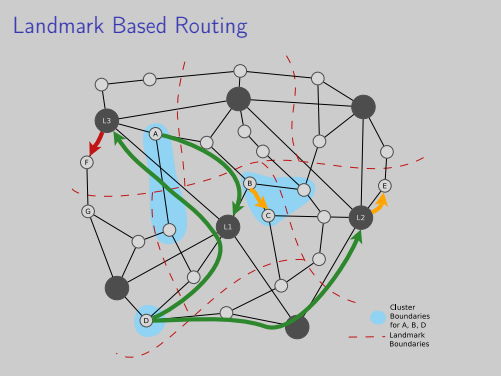
Routing Table Scalability



1. I'm interested in the rate at which routing tables increase in size with any increases in the size of the network. Particularly for the Internet. Many common routing protocols in use today are shortest-path based, including BGP. They produce routing tables which scale supra-linearly with the size of the network. So If a network grows very quickly, then router resources could struggle to keep up. The internet has at times in the past seen fast growth, and some people worry it may do so again. So it would be useful to have routing protocols whose state scaled better.

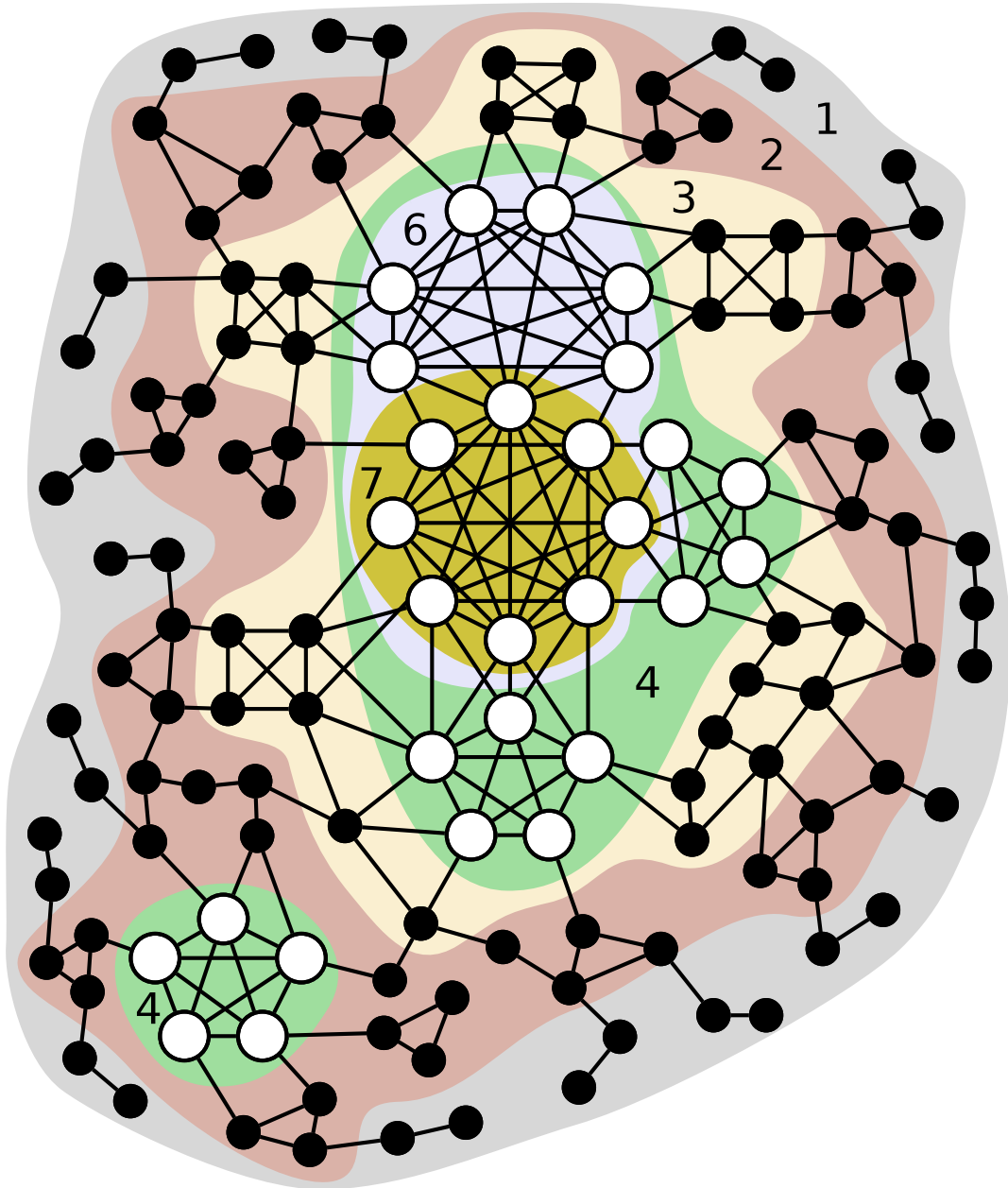
Landmark Based Routing

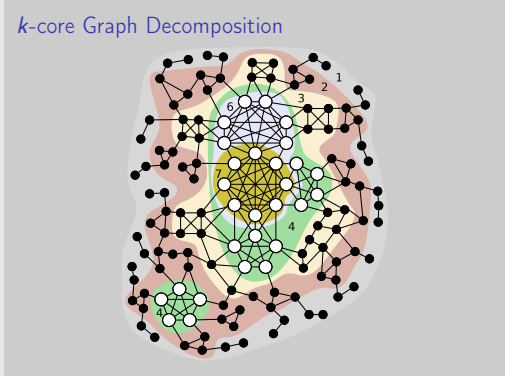




1. One way to achieve that is to do routing using landmarks. Such as the scheme described by Thorup and Zwick, and Cowen before them. The general idea is each node stores routes only for a few other nodes in a cluster around it, as well as for a select set of "landmark" nodes. Messages for any other nodes are sent towards the landmark associated with that node. The problem is there isn't yet a complete distributed, routing protocol based on this suitable for the Internet. One barrier to that is that we need a distributed way to pick or nominate the landmarks.

k-core Graph Decomposition





1. There is a graph algorithm that looks like it could help, called the k-core or k-shells graph decomposition. What this does is it assigns every node to a series of concentric shells, so that a node in a k-core must have at least k other neighbours in that core. The highest cores in a graph should be more highly interconnected and more central cores of the graph. Unfortunately, the algorithm known for this required global knowledge of the graph. There was no distributed algorithm known, that I could use to help me solve my distributed landmark selection problem.

Distributed k-core Graph Decomposition

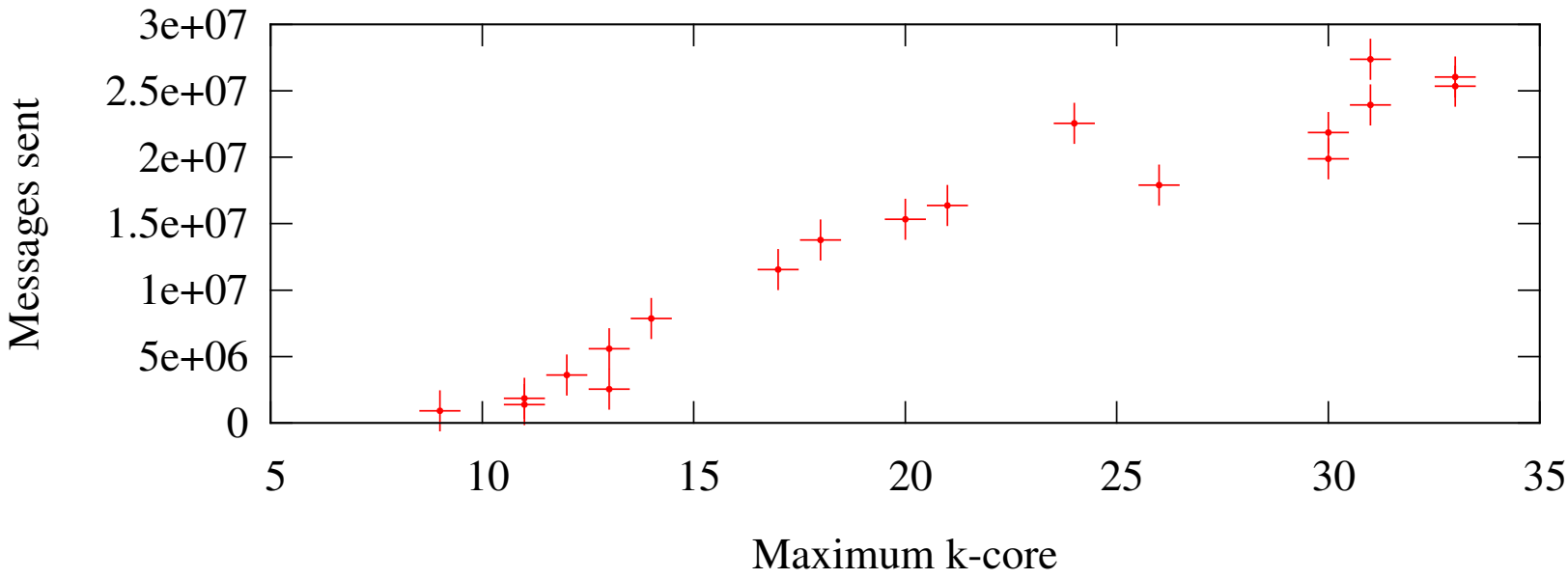
1: **for all** $n \in N$ **do**
2: $S_n \leftarrow \deg$ // initialise k-value for each neighbour
3: **end for**
4: $k_t \leftarrow \mathrm{kbound}(S)$
5: send_to_neighbours($\langle k_t \rangle$)
6: **loop**
7: $t \leftarrow t + 1$
8: **for any** $n \in N$ **do** // wait for message
9: $S_n = \mathrm{receive}(n)$
10: $k_t \leftarrow \mathrm{kbound}(S)$
11: **if** $k_t \neq k_{t-1}$ **then**
12: send_to_neighbours($\langle k_t \rangle$)
13: **end if**
14: **end for**
15: **end loop**

Distributed k-core Graph Decomposition

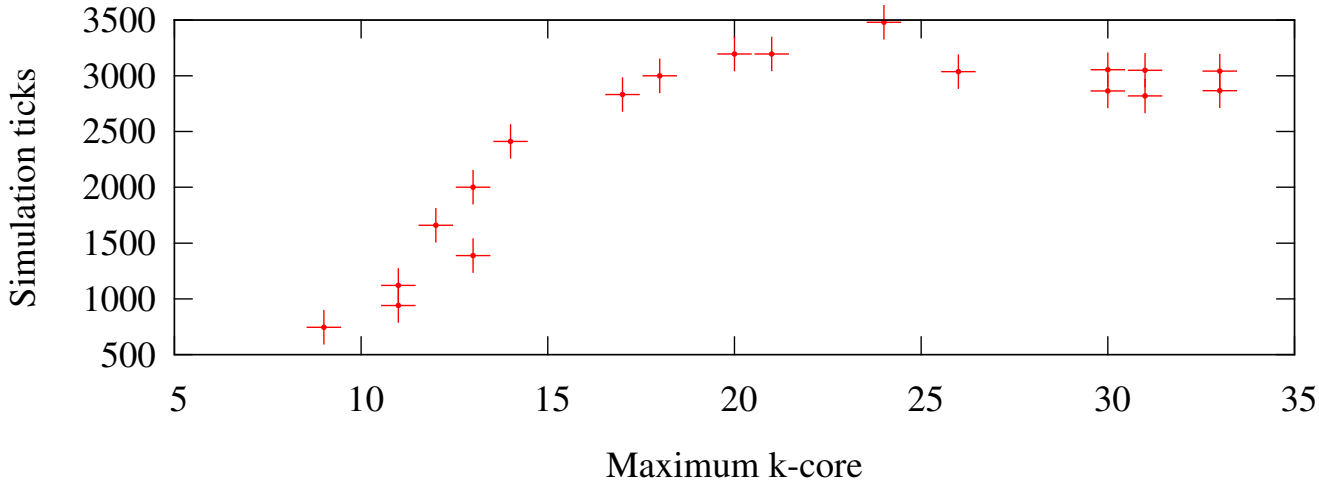
```
1: for all n ∈ N do
2:     S_n ← deg // initialise k-value for each neighbour
3: end for
4: k_t ← kbound(S)
5: send_to_neighbours(⟨k_t⟩)
6: loop
7:     t ← t + 1
8:     for any n ∈ N do // wait for message
9:         S_n = receive(n)
10:        k_t ← kbound(S)
11:        if k_t ≠ k_{t-1} then
12:            send_to_neighbours(⟨k_t⟩)
13:        end if
14:    end for
15: end loop
```

1. Our contribution then is this distributed, continuous form of the algorithm, to find the k-core decomposition of a graph or network. It works by having each node start from a clear upper-bound on their maximum k-core value, which is the node's absolute degree. Then all the nodes collaborate to progressively tighten down this upper-bound, until it converges on the correct, maximal k-core value. So each node can find its maximal k-core value, keeping no more state than the value of their neighbours. Our preliminary results suggest it scales reasonably well both in runtime and in communication overhead on Internet AS graphs. Thank you for your time.

Backup: Maximum k-core v Messages Sent



Backup: Maximum k-core v Simulation ticks



Backup: Graph size (nodes) v Messages sent