BeRGeR Bloopers

Material not included in the paper

Brown Zaz

Department of Computer Science, Kent State University, Kent, OH 44242, USA
zbrown@cs.kent.edu

Redundant routing. Without cryptography, deterministically delivering a message from s to t requires that either s and t are neighbors (the trivial case which, for the purposes of this paper, we ignore) or that there exist $2x+1$ disjoint paths between s and t. This means for any deterministic Byzantine-resistant routing algorithm to work for all pairs of nodes, there must be $2x+1$ disjoint paths between each pair of nodes in the network. By Menger's theorem, this is equivalent to requiring the network be $(2x+1)$-connected.

In spite of the necessity of $2x+1$ disjoint paths, the key insight to this routing algorithm is that packets do need to traverse these disjoint paths. If you have more than $2x+1$ packets, it is not necessary that the paths they take be pairwise disjoint; the more general condition that they are *collectively disjoint* is sufficient. Intuitively, this means that the union of the paths traversed by the nodes must contain $2x+1$ disjoint paths, but the packets do not need to follow those disjoint paths. The paths the packets take may overlap. When the paths overlap, but collective disjointness is maintained, this is called *braiding*.

It is easier to find collectively disjoint paths than pairwise disjoint paths because collective disjointness (CD) is less restrictive than pairwise disjointness (PD), as can be seen from their respective termination conditions:

$$\exists i,j \quad : \quad m_i = m_j, \qquad p_i \cap p_j = \emptyset \tag{PD}$$
$$\exists i,j,k,\ldots \quad : \quad m_i = m_j = m_k = \cdots, \quad p_i \cap p_j \cap p_k \cap \cdots = \emptyset \tag{CD}$$

1 Tikz diagrams

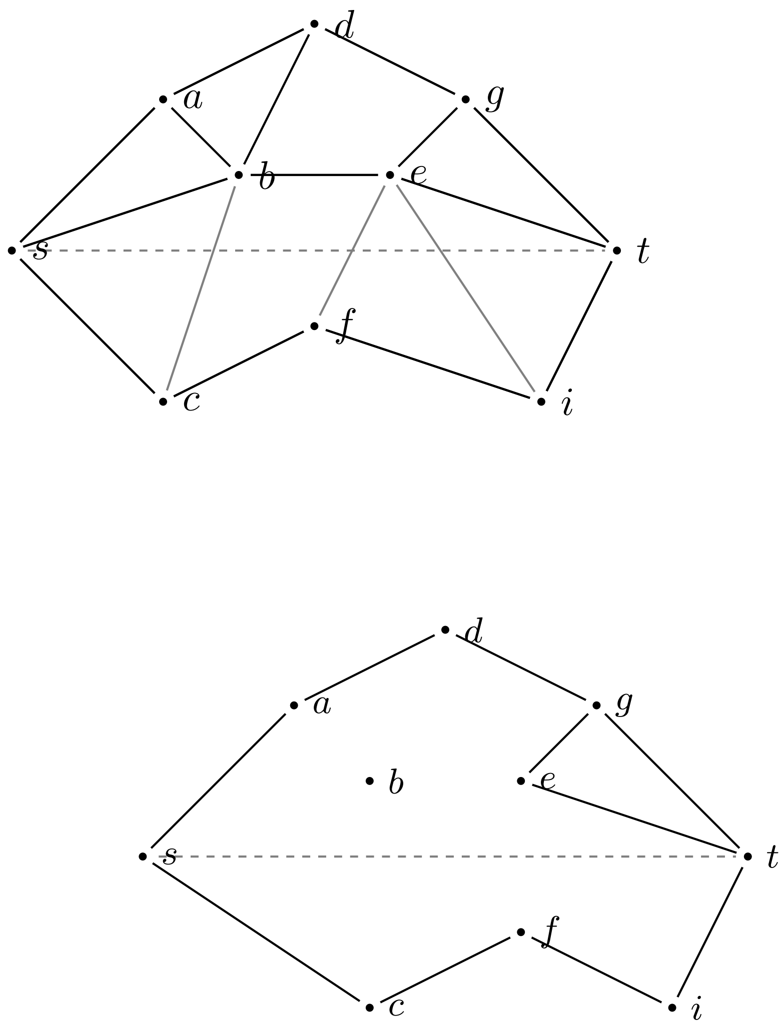
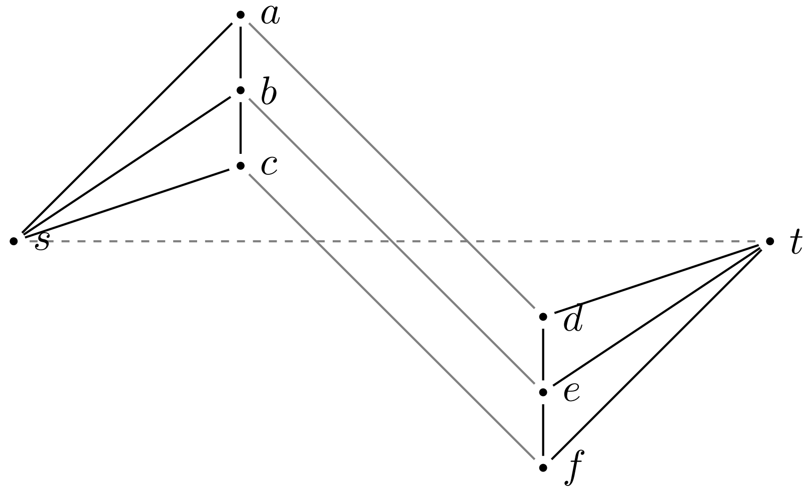
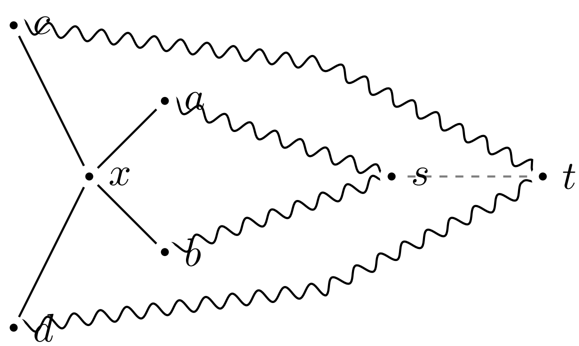


Fig. 1. Skipping b


a
b
c
s
t
d
e
f


c
a
x
s
t
b
d

2 Old Pseudocode

Algorithm 1: BeRGeR variables and functions

```
functions
nextNode(p, s, t, c, k)
    loop for i ∈ N starting with p in direction c ∈ {L, R}
        if i ≠ k and onThisSide(s, t, n, i) then
            return i
    return ⊥                                    // no next node found
```

3 Old Introduction

We have a network (graph), G, of nodes that know only the current packet, their position, and the set of their neighbors' positions. They do not have space for routing tables, nor space to store information about packets that pass through them. The goal is to route a message from s to t such that t will decode the correct message even in the presence of 1 node that behaves arbitrarily (i.e. is Byzantine).

The simplest solution without a Byzantine node would be to greedy route; at each step going to the node geometrically closest to t. However, local minima result in this naïve algorithm not reaching t. A guaranteed solution is to include draw a line through s and t (encoded in the packet), and then route along all edges of any face that intersects the $s-t$ line. This was done in Concurrent Face Routing [**?**] and is adapted here to generate 2 disjoint paths, or "cores", that form the core of our braided routing algorithm.

The state of the art in Geometric routing is GOAFR+ [**?**] where they route packets within an ellipse, increasing the size of the ellipse if progress is not made. They demonstrate practical efficiency and prove their algorithm is asymptotically optimal in the worst-case: $O(p^2)$, where p is the shortest path length from s to t, and assuming a unit-disk graph (UDG). We attempted something similar, but were stymied by networks where the external face intersected the $s-t$ line in multiple places; we may attempt this again assuming no such intersections.

Finally, [**?**] proves a correspondence between interactive consensus conditions (ICCs) and error-correcting codes (ECCs). But I believe the ICCs apply to broadcast networks (complete graphs), so are not directly applicable here.

4 Definitions

proximal sender the node the current packet was received from
$\overline{st}$ the line segment between s and t
G the graph
$G - \overline{st}$ the graph with edges that intersect $\overline{st}$ removed
superface the union of faces that intersect $\overline{st}$
green node a node adjacent to the superface. This includes s and t.
core a packet that traverses the superface using either right-hand-rule (R) or left-hand-rule (L)
braid a packet that is routed exactly as a core packet, except it skips a single node, k, indicated in the packet
braid chain a set of braids that collectively skip each node a core packet visited
matching packets packets that contain the same m, s, t

There are two kinds of packets: Cores and braids. Both packets contain:

m the message, comparable for equality
s the ultimate source of the message
t the ultimate target of the message
c the chirality of the packet: R or L

In addition, cores contain:

ℓ the set of nodes the packet has visited

And braids contain:

k the node the packet should block/skip

So cores have size proportional to the core path length, while braids are constant size ($\log n$ bits). The fact that packets contain chirality means that when a packet arrives at a node, it is not only associated with an edge, but also a face.

Delivery Condition 1 *Target receives matching core R and L packets*

Delivery Condition 2 *Target receives a core and matching braids that skip each node in ℓ*

5 Mathematical Formalism

Formally, each node, except for 1, uses the same routing algorithm, f, that takes a message, $m \in M$, the proximal source of the packet, $\mathbf{v}_{-1}$, the current node's position $\mathbf{v}_0$, and the set of the current node's neighbors' positions $\mathbf{V}$ and returns a set of (possibly modified) packets along with the node to pass each packet to:

$$f : m, \mathbf{v}_{-1}, \mathbf{v}_0, \mathbf{V} \mapsto \{(m \in M, \mathbf{v}) : \forall m, \mathbf{v} \in \mathbf{V}\}$$

Alternatively, you can imagine that there is a copy of G, G_m, for every m. We will call the union of these G_M. Changing a message from m_1 to m_2 is now expressed as moving it from G_{m_1} to G_{m_2}. Now f forms a path-sensitive directed graph:

$$f : \mathbf{v}_{-1}, \mathbf{v}_0, \mathbf{v}_M \mapsto \mathcal{P}(\mathbf{V}_M)$$

Which is a directed graph, modified to be sensitive to the proximal source of the packet. More formally, a directed graph, G, is defined by an ordered pair of a set of vertices and set of arches (V, A), and an arch is defined as an ordered pair of vertices. I define a path-sensitive graph similarly, but with A being an ordered *triple* of vertices.

It is critical that this graph is acyclic, otherwise our algorithm will not terminate.

Our solution only uses very weak assumptions. E.g. We only use the packet for the $s-t$ line, chirality (whether the packet will take the next available R node, or L node), and a single excluded node. So their may be some way to simplify the above to a directed graph. We only use the positions of nodes (along with the $s - t$ line encoded in the packet) to exclude neighbors on the other side of the source-target line. So we reduce all the GPS information down to 1 bit per edge.