ICE: A TESSERAL P2P SUBSTRATE
TECHNICAL REPORT 617

DANIEL CUTTING AND BJÖRN LANDFELDT
SCHOOL OF INFORMATION TECHNOLOGIES
THE UNIVERSITY OF SYDNEY

AARON QUIGLEY
SCHOOL OF COMPUTER SCIENCE & INFORMATICS
UNIVERSITY COLLEGE DUBLIN

AUGUST, 2007
ICE: a tesseral P2P substrate

Daniel Cutting, Björn Landfeldt
School of Information Technologies
University of Sydney, Australia
{dcutting,bjornl}@it.usyd.edu.au

Aaron Quigley
School of Computer Science and Informatics
University College Dublin, Ireland
aquigley@ucd.ie

Abstract

This technical report describes a novel peer-to-peer (P2P) substrate called ICE. It offers two features upon which a variety of P2P applications can be founded: hierarchical tesseral addressing of a logical surface divided between peers; and efficient amortised routing for delivering messages to multiple regions of the surface. This paper details the ICE philosophy and algorithms used, and presents results of experiments exploring its routing ability. ICE was originally developed as the basis of SPICE, an implicit group messaging (IGM) implementation using a tag-based modelling language.

1 The ICE philosophy

This report introduces a novel substrate called ICE based around tesseral addressing and an efficient amortised multicast routing algorithm. ICE is a useful fundament for building P2P systems and has just two functions: to organise the peers into a structured overlay over the physical network; and to provide an efficient routing algorithm for delivering messages from source peers to multiple destination peers. Figure 1 illustrates the layered approach.

Peers are organised on the $d$-dimensional surface of a $d$-torus. The entire surface is claimed by peers; there are no “holes”. Each peer “owns” an exclusive region of the surface and communicates directly only with peers that own bordering regions (maintained in a neighbourhood table). Messages are routed across the surface by passing them from neighbour to neighbour in the direction of the destination. Figure 2 shows the 2-dimensional surface of a 2-torus. The projected surface is considered to adjoin continuously on all edges.

Peers become part of the surface by routing a join request to a chosen region on the surface from an arbitrary bootstrap peer. The peer that owns the chosen region handles the join request by halving its region, returning a join acknowledgement to the new peer, and informing its neighbours.

The only state information stored by an ICE peer is its neighbour table. The number of neighbours a peer has is dependent on the dimensionality of the surface. Because the surface is randomly partitioned between peers, each peer has approximately one bordering neighbour for each face of its region, or two neighbours for each axis of surface dimensionality. Thus, the storage complexity of ICE is related only to the dimensionality of the surface, i.e., $O(d)$. The number of peers does not affect the state stored by any individual peer.

The ICE surface is superficially similar to CAN [6]. However, there is a crucial difference. Unlike CAN which uses a Cartesian coordinate space, ICE surfaces use hierarchical tesseral addressing. The distinction is motivated by the intended usage: CAN is designed to support point-to-point routing whereas ICE is inherently a multipoint routing substrate.

2 Tesseral addressing

Hierarchical tesseral addressing is a compact and elegant addressing scheme that describes regions of
space instead of Cartesian points. It is particularly useful for arbitrarily decomposing a space to different granularities and is used to decompose the ICE surface.

The general idea is to efficiently decompose a plane by subdividing it into a regular tessellated pattern, so that each element can completely contain another such division. Squares and triangles have this property. Figure 2 illustrates a hierarchical tesseral address space based on squares; such a structure is often referred to as a quadtree.

Hierarchical tesseral addressing has been applied to many domains including geographic data storage and querying, computer graphics and robotics [7]. It has not been used to decompose the address space of a structured P2P network, although some systems have stored such structures over existing structured networks [9]. The properties of hierarchical tesseral addressing are crucial to the routing algorithm used in ICE to deliver messages to multiple peers (Section 3).

ICE regions are addressed by strings of \( d \)-bit digits (of base \( 2^d \)). These regions are termed extents, and each digit represents a progressive index into the hierarchically addressed surface. Extents may be large and coarse-grained or extremely small and fine-grained. The depth of an extent is the number of digits in its address. Deeper extents are necessarily smaller than shallow extents. The solitary extent of zero length is the universal extent (denoted \( U \)) and represents the entire surface.

Figure 2 illustrates the addressing scheme, ordered left-to-right and top-to-bottom. For example, the address 0 specifies the top-left quadrant of a 2-dimensional surface, and address 3032 specifies a small extent towards the bottom-right corner. This mapping approach can be applied to surfaces of arbitrary dimensionality without loss of generality.

A set of extents is called a tract. Tracts can represent arbitrary regions of the surface by composing several extents, both contiguous and disparate. For example, the tract \( \{0, 2\} \) is the entire left half of a 2-dimensional surface. Every instance of a tract has a definite representation (i.e., a specific set of extents), although there are infinitely many such sets which can define the same tract. Tracts can be subtracted from one another but unlike ordinary set difference, the result is a new tract that describes the remaining region of the surface. For example, on a 2-dimensional surface, \( \{0, 2\} - \{20\} = \{0, 21, 22, 23\} \).

Hierarchical tesseral addressing has some important features that make it suitable for describing an overlay surface. Its hierarchical nature allows a deep address to naturally scale to increasingly large covering extents simply by omitting digits from its tail. Similarly, its self-similar properties means an address can easily be translated across the surface by replac-
ing its head with digits from another extent.

It is important to note that although this addressing scheme is hierarchical, there is no need for a global data structure to be maintained across peers. In particular there is no “root” peer. It simply affords compact descriptions of large and small regions of the surface, and is a convenient way for peers to store surface information and communicate with one another. For instance, the region of the surface that each peer owns is represented as a tract. Likewise, peers’ neighbourhood tables map extents adjacent to their tract to the peers that own them. The versatile hierarchical and self-similar properties of tesseral addressing are a core part of the multicast routing algorithm now described in Section 3.

3 Amortised routing

Besides tesseral addressing, the other major component of ICE is an efficient amortised multicast routing algorithm. Payloads are routed from a source peer to a destination tract by passing messages from neighbour to neighbour. All peers that own tracts intersecting the destination tract receive a copy of the message.

3.1 Point-to-point routing

ICE can emulate point-to-point routing by delivering to a very small destination tract that is covered by the tract of a single peer. Messages are routed geometrically across the surface between neighbouring peers. At each hop, peers forward the message to a neighbour that owns an extent nearer to the destination. Figure 3(a) shows an example of a message routed across a 2-dimensional surface.

For the purposes of finding a neighbour closer to the destination, a distance metric between extents is required. This is achieved by mapping extents from their tesseral addresses to bounding hypercubes in a Cartesian space. To do this, the surface is assumed to be wholly contained within a unit hypercube, within which extents are converted to sub-hypercubes. The distance between two extents is defined as the length of the minimum interval connecting their bounding hypercubes. For example, the distance between extents 3032 and 23 on a 2-dimensional surface is 0.125 (see Figure 2).

The message complexity of point-to-point routing depends on the dimensionality of the surface and the number of peers. An equally partitioned surface has \( n^{\frac{1}{2}} \) distinct peers arranged along each edge of the surface. The longest route a message can take is from a corner of the surface to its centre (due to the surface wrapping on all sides). The surface is “quantised” by the peers so the message takes a Manhattan (“city block”) path. Point-to-point routing over the ICE surface is thus \( O(dn^{\frac{1}{2}}) \) overlay hops. Figure 3(b) confirms this in simulation by routing messages between two random extents. As expected, higher dimensionality requires fewer hops to route between two parts of the surface.

In practice, actual routing performance can be improved by considering the physical link latency between neighbours when selecting the next hop, in addition to the distance from the destination. The “best” hop is chosen to be the one with the greatest distance progress to link latency ratio which tends to favour low latency links, when all else is equal. Listing 1 shows this fragment of the ICE routing algorithm. The technique is one of the routing optimisations used in CAN [6], but other routing possibilities exist. For example, peers could attempt to minimise total load on neighbours over time, avoid known malicious peers, or monitor the reliability of neighbours in order to bias route selection.

3.2 Multipoint routing

The ICE routing algorithm is not restricted to point-to-point routing; it delivers copies of a message to all peers with tracts intersecting a destination tract. The destination tract may be a small contiguous region of the surface but this is not a requirement. A tract can specify any region of the surface whether contiguous or not. There are times when it is useful for a message to be delivered to disparate parts of a surface. For instance, a resilient storage layer built atop ICE could use the routing algorithm to copy data to multiple regions of a surface.

The algorithm is given a destination tract and a
Routing across a 2D surface.

(a) Routing across a 2D surface. (b) Point-to-point routing is \(O(dn^{\frac{1}{2}})\) overlay hops.

Figure 3: ICE point-to-point routing.

Table 1: Object structure for ICE algorithms.

(a) An ICE Peer object.

<table>
<thead>
<tr>
<th>ICE Peer</th>
<th>tract</th>
<th>neighbourhood</th>
<th>handled?</th>
</tr>
</thead>
</table>

(b) A Route object.

| Route | destination_tract | branch_factor | payload |

Listing 1 The “next hop” fragment of the ICE routing algorithm.

```python
def next_hop(p, tract):
    current_dist = p.tract.distance_to(tract)
    hops = p.neighbourhood.distances_to(tract)
    progress = hops.map do |neighbour, next_dist|
        {'neighbour' => neighbour,
        'progress' => (current_dist - next_dist) /
        p.neighbourhood.latency_to(neighbour)}
    end
    max_progress = progress.max { |x,y| x['progress'] <=> y['progress'] }
    max_progress['neighbour']
```

4
payload. A peer routes the message by forwarding it to the neighbouring extent that is closest to any part of the destination tract. At each hop, the peer handling the route delivers the payload to the application layer if its tract intersects the destination tract. Its tract is then subtracted from the destination, reducing the total region left to visit. The remaining tract is then recursively visited in the same way.

When a tract is not contiguous, it is quicker to route separate copies of a message to each region instead of visiting each in turn. However, routing separate copies of a message from the source is inefficient if part of the route is the same for each copy. Because Ice is a structured overlay and the surface is a geometric construct, the routing algorithm clusters parts of the destination tract based on their direction away from the source in order to amortise the routing cost. If several lie in the same direction away from a source, it is sufficient for a single copy to be routed that way.

This technique is recursively applied. Not only is the destination tract clustered from the initial source, it is clustered again at each hop. This results in messages taking a tree-like route from the source to all peers with tracts that intersect the destination tract. Initially the route has few branches but as the message approaches individual peers it branches as necessary to balance the total number of messages against direct individual routes. Figure 4 shows an example of amortised routing delivering a message to a highlighted destination tract (comprising two distinct regions).

A divisive hierarchical clustering algorithm is applied to the remaining destination tract at each hop resulting in a set of clustered sub-tracts. Each of these clusters is then individually routed a copy of the message. Thus, the message branches when there is more than one cluster but continues in a point-to-point fashion otherwise.

Divisive clustering transforms a single set of objects into a number of clustered sets. It works by first finding the maximum “distance” between two elements in a set and creating a cluster for each. Each remaining element is then assigned to the cluster to which it is closest. This continues until each element is in its own cluster, or the maximum distance between every element is below a threshold. For example, the following set of five numbers is clustered according to their difference, resulting in first level clusters of two and three elements, and so on until each is individually clustered.

<table>
<thead>
<tr>
<th>13568</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
<tr>
<td>568</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

Recall that a tract is simply expressed as a set of extents; the clustering algorithm groups these extents according to the angular difference between their centres as measured from the point of view of the current peer’s tract. Those with an angular difference larger than a threshold are placed in separate clusters. This threshold is called the branch factor, expressed in radians and taking a value in the range $0–\pi$. Due to parallax, distant extents are more likely to be clustered together than nearby extents.

A branch factor of 0 means messages are never amortised; every extent in a destination tract is considered its own cluster and individually approached using point-to-point routing. A value of $\pi$ means the
entire destination tract is always treated as one cluster such that the message never branches but visits each extent in turn. The advantage of a high branch factor is that a minimal total number of messages are needed, which in turn reduces the incoming and outgoing load on peers and network links. The drawback is that messages take longer to reach their destinations on average, because more circuitous routes are taken.

Listing 2 presents the complete ICE routing algorithm, building upon the “next hop” fragment implemented in Listing 1. Since ICE is designed as a general basis for P2P applications, it provides callback hooks that an application can use to modify the default multipoint routing behaviour. In particular, two callbacks may be implemented by an application: callback_deliver (Line 9 of Listing 2) and callback_branch (Line 20). These are called prior to the delivery of a payload to an application, and prior to forwarding a payload to a cluster of the destination tract. They may be used to alter the payloads that are delivered, or veto them altogether. The default implementations (Lines 32 and 38) behave as described above, delivering the same payload to every peer in a destination tract.

The effect of surface dimensionality and branch factor on multipoint routing are investigated via simulation. 1000 messages are routed from random sources to random destination tracts intersecting \( \approx 500 \) peers of 1000 on an ICE surface. The total number of messages needed for each delivery and the average number of hops to each destination peer per message are measured.

Figure 5(a) shows that the branch factor has a very significant effect on the total number of messages. By increasing amortisation of packets (with a higher branch factor), the number drops to approximately 1000 irrespective of surface dimensionality, just twice the minimum number of messages needed to directly contact each destination peer from the source. Increasing the surface dimensionality generally increases the total number of messages because there are intuitively more directions in higher dimensions and destinations are less likely to be clustered. However by increasing the branch factor beyond approximately \( \frac{\pi}{3} \) the total number of messages can be reduced even in these higher dimensions because the algorithm is more permissive in what constitutes a “similar” direction.

Higher dimensionality also dramatically reduces the average number of hops to destinations (Figure 5(b)) because the number of hops between two parts of the surface is reduced (as shown previously in Figure 3(b)). A lower branch factor also delivers messages in fewer hops since they do not follow longer generalised routes to many destinations. A branch factor of less than \( \frac{\pi}{3} \) tends to produce the most direct routes to destinations.

Based on these results, a surface dimensionality of 3 and branch factor of \( \frac{\pi}{3} \) offer the best compromise between low total messages and average hops to each destination.

4 Overlay mapping

It is theoretically appealing to order or arrange peers randomly in the address space of a structured overlay, as uniform randomness simplifies design and permits assumptions of system behaviour and costs at the overlay level. However, such randomness does not consider physical network locality. Because neighbouring peers may be physically separated by intercontinental distances, such overlays may exhibit appalling performance. Therefore, it is beneficial for the overlay surface to map well to the physical underlying network, although this must be tempered by the desirable properties of random placement. Many systems offer functionality to aid creation of overlays that map well to physical networks. These are generally based upon measuring the latency of connections to other peers, or to a set of well-known “landmarks” around the network and include the ping-based Vivaldi [2], Madhyastha et al’s traceroute-based approach [4] and CAN’s landmark binning scheme [5]. Any of these approaches may be applied to the construction of the ICE overlay.
**Listing 2** The ICE routing algorithm.

```python
# Peer p routes 'route'.
def handle_route(p, route):
    # Deliver the payload to this peer if it's in the destination tract.
    if route.destination_tract.intersects?(p.tract) and
        not p.handled?(route):
        # A callback allows an application to alter what payload is delivered to the peer, and what continues to be routed to the remainder of the destination tract.
        route.payload = callback_deliver(p, route) do |deliver,payload|
            p.deliver(p, deliver,payload)
        end
    end

    # Subtract this peer's tract from the destination tract.
    remaining_tract = route.destination_tract - p.tract
    unless remaining_tract.empty?
        # Cluster the remaining destination tract.
        clusters = p.tract.cluster(remaining_tract, route.branch_factor)
        # A callback allows an application to alter what payload is routed to each cluster.
        callback_branch(p,route,clusters) do |cluster_payload,cluster|
            route.dup = route.dup
            route.dup.tract = cluster
            route.dup.payload = cluster_payload
            hop = next_hop(p,cluster)
            p.send(hop,route.dup)
        end
    end
    p.handled(route)
end

# By default, don't alter the payload.
def callback_deliver(p,route)
    # Yield to Ice to deliver the payload.
    return route.payload # Return the payload unaltered.
end

# By default, deliver the same payload to each cluster.
def callback_branch(p,route,clusters)
    clusters.each { |cluster| yield route.payload,cluster }
end
```
5 Self-stabilisation

The unreliable nature of peers in a P2P system guarantees that some peers will unexpectedly fail or leave the system over time. When designing a structured overlay such as the ICE surface, it is important to consider how it will cope with structural damage when it is encountered and how it will repair itself despite peer flux. ICE does not inherently support data storage — it is implemented by higher layers if needed — so the only impact of a failed peer is routing over the surface. To ensure messages are not interrupted by holes in the ICE surface (unclaimed tracts), the routing algorithms must be capable of routing around failed peers. Additionally, peers should detect and repair holes. This section sketches how the ICE surface can be made “self-stabilising”.

Self-stabilisation [8] is the process of converging a system to a stable state from an arbitrary starting state. This can be expressed as a set of invariants and rules that move the system state towards satisfying these invariants. In ICE, a stable surface is defined by three invariants. Firstly, the entire surface must be claimed by peers. Secondly, the tracts claimed by peers must not intersect. Thirdly, every extent adjacent to a peer’s tract must be described by exactly one mapping in its neighbourhood table.

ICE is suited to the correction-on-change and correction-on-use approach to self-stabilisation [3]. In this technique, problems are corrected as they are detected during normal routing operations, rather than through the use of periodic beacons. This greatly limits the amount of maintenance traffic required and is especially applicable to ICE because the failure of a peer affects only $O(d)$ neighbours.

Correction-on-use dictates that each message routed across the surface includes maintenance information about intermediate source and destination peers. This allows peers to verify that their neighbourhood state agrees with the sender’s. Deterministic agreements can ensure that any conflicts arising (such as two peers claiming intersecting extents) are resolved. Furthermore, the very act of receiving a route from a peer indicates the sender’s existence and allows the recipient to ensure neighbourhood information is correct.

When routing messages, peers will occasionally detect failed peers. In these cases the message must be alternately routed. A message may be greedily routed around the periphery of a hole on an ICE surface by probing the “next best” neighbour, or more general techniques from sensor networking or geocasting.
may be applied, such as the FACE protocols [1]. In addition to routing around holes, the correction-on-change protocol repairs the hole by deterministically assigning it to a new peer. Detections of the hole by different peers routing messages result in a deterministic marshalling peer being informed. This peer can then claim the hole itself, or delegate it to a neighbour. Finally, update information can be delivered to neighbours so they can update their neighbourhood tables. If a marshalling peer has also failed, the technique may be recursively applied.

It is clear from this sketch that implementing self-stabilising algorithms is not problematic, but note that a stabilised Ice surface does not negate the need for applications using Ice to employ their own data integrity algorithms. Ice’s only guarantee is to route messages to all peers within a destination tract. Any application-specific stabilisation must build on this.

6 Conclusion

This report has described the Ice surface, a novel structured overlay network design based on hierarchical tesseral addressing and an efficient amortised multicast routing algorithm. Ice is a very lightweight basis for building more complex P2P systems. Its only functionality is to maintain a logical surface of peers, and allow routing of messages to arbitrary regions of this surface.

Acknowledgements

The authors would like to acknowledge the support of the Smart Internet Technology CRC.

References


