The following notes are based on the page numbers on draft-ietf-list-nexagon-36 that expires on September 10, 2022.

This review was done by the H3 Steering Committee, composed of David Ellis, Nicholas Rabinowitz, AJ Friend, and Isaac Brodsky.

Summary

We are happy to see standards using H3 as the spatial indexing system being proposed. We believe H3 is well suited to use in the mobility space, and in general it seems like an excellent choice for a mobility edge network using the LISP protocol. More detailed feedback is included below.

Parent-child Relationships in H3

On page 11 it is suggested that multicast subscription is to a singular coarse-resolution cell (h3.rB) based on the parent-child relationship with the high-resolution cell (h3.rS) the client is located within. There are several issues with this:

- First, the parent-child relationship is a simple tree based on approximate containment, which works decently well between neighboring resolutions, but becomes less-and-less contained as the delta in the resolutions increases, as can be seen here.
- Second, there will be an “edge effect” as the client in question moves towards the edge of the coarse-resolution cell as the majority of the data in the feed will be “behind” it.

It would be more useful to keep the client close to the center of its data feed by instead subscribing to a kRing of hexagon IDs centered on its location rather than a singular hexagon. Even just a kRing of 1 would reduce this problem significantly, and there would be a net-zero impact on data volume if the coarse-resolution hexagon ID was reduced in conjunction with the kRing to produce an equal area coverage (h3.rB == kRing(h3.r(B-1), 1) == kRing(h3.r(B-2), 3), etc).

Distributed Backend Architecture

On page 4 you can find the only explanation of how the backend for this protocol should be developed, with a reference to a “density-factor” fudge factor and apparently explicit allocation of the coarse-resolution hexagons (h3.rB) to specific servers based on data volume. This could work for data ingest, but there is no explanation on how the subscribers will get all messages when they are sharded between servers, nor what would happen if the subscribers cannot ingest the total data volume in question. We believe this is a suboptimal architecture, having prior experience at Uber dealing with something similar, and are willing to provide an alternative, but given the small amount of space in the IETF dedicated to the backend implementation, we aren’t sure if this is the point of the RFC.

Tracking Location Accuracy

On page 9 the unicast detection upload logic is described and the precise message payload outlined. We noticed that location accuracy estimation (from the combined uncertainties of the GPS device and the offset estimation) are not included, only the high resolution H3 cell ID (h3.rS). Due to later sections dedicated to the importance of clients being able to filter out poor-quality detections from their subscriptions, we believe the estimated accuracy radius should be included in some form. In the appendix at the end of this review we have a set of proposals of varying levels of overhead that could add to or modify the byte layout of the header format.

Anonymity and Trust

On page 15 and page 16, there are two contradictory statements surrounding the MobilityClientEIDs. On page 15 it is said that clients could learn to trust or distrust data sources by MobilityClientEID and on page 16 it is said that clients would remain anonymous because their MobilityClientEIDs would shift frequently. It would not be possible to satisfy both – if the trustworthiness of the data source is resolved, then it would be possible to make a solid guess as to its current location and the path it has taken, while if the MobilityClientEIDs shift frequently enough to prevent this, it isn't possible to determine if the data in question is valid or spoofed in some way or another.

We don't consider this a solved problem so this is not a hard criticism, but these two sections should be reworded to be more explicit about the trade-off here. For particular, tightly-regulated implementations of this protocol, however, the closest thing to a solution would be the TPM system Microsoft has developed, which would allow a central authority to verify the GPS and mainboard firmware are signed builds known to the central authority, and then configure the clients to trust all data streamed to them as all clients are verified during the login process described on page 5, and a TPM-based handshake could be employed.

Terminology

On page 1 (and several other pages) the terminology "nexagon" is used without explanation. What does this mean?

Appendix: Possible Location Accuracy Estimation Solutions

GPS accuracy is usually represented as the radius of a circle around the GPS point representing a 95% confidence that the true position is somewhere within that circle. Consumer GPS, especially on moving vehicles, is not very accurate, and the detection translation offset would have its own error level, which would further expand this radius.

This could be transmitted through as an IEEE754 floating point number, but that makes the accuracy as large as the H3 index itself, which is not super useful. For the purposes of deciding on the usefulness of an observation, though, a couple of coarser options are possible.

1. A single byte could be added to the header to indicate the kRing that best covers the observed point encoded in the high resolution cell (h3.rS), allowing 0-255 rings with a from 1 times the high resolution cell radius to 416 times the high resolution cell radius.
2. A nibble could be added in some set of reserved bytes to include the approximate H3 cell resolution that the error radius best represents. This provides 16 levels of granularity, which is worse than the first option, but more closely maps to the H3 internal data structure and opens up a third possibility.

The third possibility would require creating a new H3 cell type, so unless you find the use-case very advantageous, we would not implement it. In the H3 cell byte format, the nibble for the resolution is effectively redundant and provides a kind of error detection to the cell ID. The 16 resolution bit triplets encode all of the data necessary to determine which cell and what resolution the cell is, based on which bits are enabled or not.

The proposed format would use the resolution bits as the accuracy approximation as laid out in the second suggestion, while the resolution 0 to 15 bit would encode the actual hexagon resolution and position. Likely you would use resolution 15 at all times in this mode as that would produce the least error in a (lat, lng, err) -> this new cell type -> (lat, lng, err) conversion, but the cell format itself would not need to limit itself in that way.

With this, you would presumably get a form of the accuracy error “for free” in the current draft protocol, and you could even keep the current form with no error information for observations unwilling or unable to provide an error estimate, by just using the standard cell type for those observations. Similarly, clients could easily discard this error information internally and transform the cell back to the standard type if they are unable or unwilling to deal with anything other than point data. It would require this new cell type to be created in H3 and all clients to at least tacitly be aware of this, but could deal with this portion of the RFC that otherwise feels incomplete.