Previous Designs and Implementationssec:previous-designs

Before working on ShareTrace, I did not have experience developing distributed algorithms. The approach proposed in ch:risk-propagation is my fifth attemptIn completing this thesis, I have learned two important rules: define requirements clearly and recognize when those requirements are (not) satisfied. The time and effort needed for this thesis could have been significantly reduced had I adhered to these two simple rules.t defining a performant implementation of risk propagation that satisfies the requirements of being decentralized and online. The prior four attempts offered valuable learnings that guided me toward the proposed approach. In fact, the approach defined in sec:projected-subgraphs is the topic of published work Tatton2022. To document my efforts in developing this thesis, prior designs and implementations are provided in this appendix.

Thinking Like a Vertexsec:giraph

The first iteration of risk propagation Tatton2020 utilized Apache Giraph Giraph2020, an open-source version of the iterative graph-processing library, Pregel Malewicz2010, which is based on the bulk synchronous parallel model of distributed computing Valiant1990. Giraph follows the "think like a vertex" paradigm in which the algorithm is specified in terms of the local information available to a graph node McCune2015.

Risk propagation was implemented as defined in Ayday2020,Ayday2021, using the factor graph representation of the contact network. However, because the Google/Apple API does not permit remotely persisting ephemeral identifiers, the implementation assumed that user geolocation data would be analyzed to generate the factor nodes in the factor graph (sec:location-based). fig:aws-architecture describes the high-level architecture. Callouts 1, 2, and 4 were implemented using a fan-out design pattern in which a ventilator Lambda function divides the work amongst worker Lambda functions.

figure[htbp] [width=]aws-architecture [ShareTrace batch-processing architecture]ShareTrace batch-processing architecture. (1) An AWS Lambda function retrieves the recent risk scores and location data from the Dataswyft Personal Data Accounts (PDAs) of ShareTrace users. Risk scores are formatted as Giraph nodes and stored in an Amazon Simple Storage Service (S3) bucket. Location data is stored in a separate S3 bucket. (2) A Lambda function performs a contact search over the location data and stores the contacts as Giraph edges in the same bucket that stores the Giraph nodes. (3) Amazon Elastic MapReduce (EMR) runs risk propagation as a Giraph job and stores the exposure scores in an S3 bucket. (4) A Lambda function stores the exposure score of each user in their respective PDA. fig:aws-architecture

ese two simple rules. a