During the COVID pandemic, tracing exposures to infection quickly and accurately has become a critical task to limit both the speed and the extent of viral spread within communities. Several existing systems have already taken advantage of smartphone technologies to automate the process of identifying and responding to exposure events, with various tradeoffs between each system's scope and functionality and its performance and protection of users' privacy. Our goal is to design a better exposure tracing system that meets the requirements of a university setting (Section IIIC).

Here, we briefly highlight the properties of existing exposure tracing systems, define the key requirements and modules of our system, and examine its desired behaviors in a couple of use cases. We end with outstanding questions about the system specifications.

Existing Systems

Existing systems contend with trade-offs between functionality, scalability, and privacy. Some existing systems collect a large amount of personal information, including geolocation both before and after infection or exposure, and perform most of their processing on a central server (Sections IIA and IIB). While this allows for the most direct medical and public health responses, this also involves a more serious breach of user privacy.

On the other hand, the existing system that supports the greatest level of privacy faces limitations in terms of accuracy and scalability (Section IIC). Most of the system is decentralized, with users voluntarily self-reporting positive test results, so information may be incomplete. Additionally, since almost all processing is done entirely on individual phones, every phone must process all positive infections within the system. This can be expensive both for the phones and for the networks that need to broadcast these infections to every device.

Key Requirements and Modules

Our exposure tracing system must support several key features (Section I). First, it must identify contact events, where individuals are close together for an extended period of time. Second, it must notify those infected, as well as the system, of positive test results, providing support as necessary. Third, it must determine individuals who were exposed to an infected individual and notify them so they can begin isolation. Fourth, it should provide information for medical support and public health needs. Lastly, it must protect user privacy.

There are several interacting modules within our system. First are the different devices (Section IIIB) — a central server, a set of Wifi routers, and a large number of individual smartphones — which can communicate using a variety of methods including wired networks (server-to-router), Wifi (router-to-phone), and Bluetooth Low Energy (phone-to-phone). From a user perspective, we can also consider the people involved: individuals using the app (infected, exposed, or neither) and the central organization running the service (Section IIIA). Individuals will interact with the system through their phones, while the central organization will have access to the central server.

Another important consideration is the kind of data that is being generated and where it is being processed and stored within the system. Physical interaction networks, including all contact events and possibly geolocation, are obtained by phones and routers through BLE and Wifi signals. On the other hand, social networks, including information about shared classes and living units, as well as contact information, are provided by the central service. A key part of the design, therefore, will be to decide where and how to link these disparate networks. This will have important implications for performance and privacy.

Additionally, communication protocols will need to be defined between the different devices. A consideration here is how different devices are named — whether each phone has a stable ID over time or one that changes, and who controls the information needed to map anonymous IDs to individual users.

Properties and Use Cases

First, all event determinations and notifications must be *scalable* and *performant* — they must not be too expensive either for individual phones or for the networks handling communications. For example, the system must be able to handle a large number of positive cases arising at one time, whether these are spread throughout the community or concentrated within a cluster. In this case, a large number of exposure events may need to be determined and notifications of infection or exposure made. The system must be able to handle this situation in an *accurate* and *timely* manner without being overwhelmed, promptly notifying those at risk or in need of support (Section IIIA and IV). This is crucial for the system to fulfill its job as an exposure tracing service.

Second, the system must protect user *privacy* by sharing the minimum amount of information needed to be useful (Section IID and IIIA). For example, data on contacts and geolocation of a person may not be relevant until they become associated with an infection or exposure. Privacy is an important concern for our system.

Third, the system should support *public health* measures. For instance, suppose the testing center where everyone goes to get tested turns out to be a site of transmission. We would want our system to help us identify this risk so that better health measures can be put in place. This function would be secondary to the primary goal of notifying exposed individuals.

Key Questions

I pose the following questions in regards to the design:

- What is the difference between isolation and quarantine? From the DP spec, it sounds like those who are infected go into quarantine, and those who are exposed go into isolation. Does that have any practical implications for how our system should treat these individuals? For instance, what kind of useful and supportive information is needed for infection and exposure?
- Will we ever be concerned with two or more degrees of separation on the interaction graph, or only with direct contact events involving a confirmed positive case?
- Where are the personal records stored? Are they located on the server or abstracted out to a service? How would we access this information?