Digital Contact Tracing : Software Architecture

Dibyendu Baksi, PhD*

***Abstract*— The covid-19 crisis is providing a lot of impetus to the search for innovative technological solutions to solve major problems of tracking and containment of the pandemic. The major cornerstones of testing, isolation, contact tracing and quarantine are well understood and agreed upon at a general level. In this paper, the software architecture for implementing successful automated digital contact tracing applications is elaborated. The goal of contact tracing is to proactively identify the infection chain of the population including asymptomatic people yet to be tested positive, i.e., to avoid asymptomatic people from spreading the disease without any intention. The entire ecosystem of contact tracing is explained so that the real challenges of integrating the key healthcare components are appreciated.**

***Index Terms*—Contact tracing, public health, surveillance, software architecture and UML.**

***Impact Statement*— Digital contact tracing is the key to get a good idea of how the current covid-19 pandemic is spreading so that appropriate control measures can be taken. A software architecture viewpoint is the place to start to understand the challenges for all potential implementors so that the entire scope of end-end requirements is addressed.**

I. Introduction

As the SARS-CoV-2 pandemic unfolds and drastic control measures to limit the spread are taken, the two main goals are the same in front of all nations.

- maintain low case numbers of newly infected population so that healthcare systems are within capacity
- relax restrictions in a phased manner to a return to normal life to minimize the socioeconomic toll

The cost benefit and economical aspects of a complete lockdown versus aggressive reopening vis-a-vis saving lives versus wrecking economies will vary depending on the context [1]. A trade-off analysis of the long term dynamics of applying different policies in the framework of Control theory from engineering is available in [2]. Asymptomatic transmission is quite common in Covid-19 and there are reports estimating the percentage to be significant [3]. This is a major containment problem with COVID-19 as infected cases can transmit the virus 1-3 days before they develop the symptoms prior to falling ill. *Contact tracing seeks to uncover the social network graph [4] of contacts of an infected person so that pre-emptive actions can be taken.*

Availability of user-friendly solutions requires precise specification of the problem leading to design options that can accommodate all the aspects. There are two ways of doing contact tracing.

- Interviews: Information of disease outbreaks with confirmed cases and possible cases are gathered using manual (i.e., interviews) or other means from test labs etc. The limitations are time and resources to conduct them and lack of scalability. The ability of the person being interviewed to remember and recall the details regarding the proximity of the contact is another major constraint.
- Digital proximity: Identifying people who come close using cellphones and bluetooth network, is the main idea behind digital contact tracing. Bluetooth low energy network is used to record cellphones that were in proximity in terms of variables such as anonymized temporary identifiers, distances, timestamp, duration etc. There are lots of options for implementation of such an idea along many angles such as user location tracking, privacy concerns related to data elements being tracked and how alerts are delivered.

Traditional ideas and practices of contact tracing involve a centralized public health tracking or surveillance system. In this paper, the focus is on contact tracing using current digital technology and state-of-the-art software design practices to maximize the exploration of the underlying exposure network graph of possibly infected persons with minimal invasion of privacy. One of the goals of this communication is to demystify the “behind-the-scene” details of contact tracing applications so that the responsible party (e.g., local, state or federal public health agencies) can make a judicious choice regarding which options to choose from in searching for a real solution. In other words, what

Dibyendu Baksi is an independent researcher in Lawrenceville, GA, USA. (correspondence dbaksi@gmail.com)

one sees in a contact tracing application on a smartphone is only a piece of the puzzle behind a successful end-end solution.

II. ARCHITECTURE

There is often some confusion in terms of what exactly is being offered by a contact tracing application. To validate and ensure an end-to-end solution, it is important to understand the trade-offs, overlapping areas and business capabilities. It is critical to define the overall architecture of something important like digital contact tracing before a solution is crafted. Architecture is defined as a 3-tuple, i.e, {*Components, Connections, Configuration*}[5]. The following sections provide the two major configurations (centralized and decentralized) of the architecture with a detailed view of the components and their connections or protocols.

The detailed design of a contact tracing application requires a few components beyond what one can view in the app itself. The success of contact tracing is critically dependent on all these components to work seamlessly with agreed upon messaging protocols and data standards. The main question around the minimum amount of data needed to alert someone who came close to an infected person sparks heated debates around what information to collect and who triggers the event. The answer to the question also provides the extent to which epidemiological research can be conducted to decipher the evolution of the spread of the disease in terms of the underlying social network graph [4].

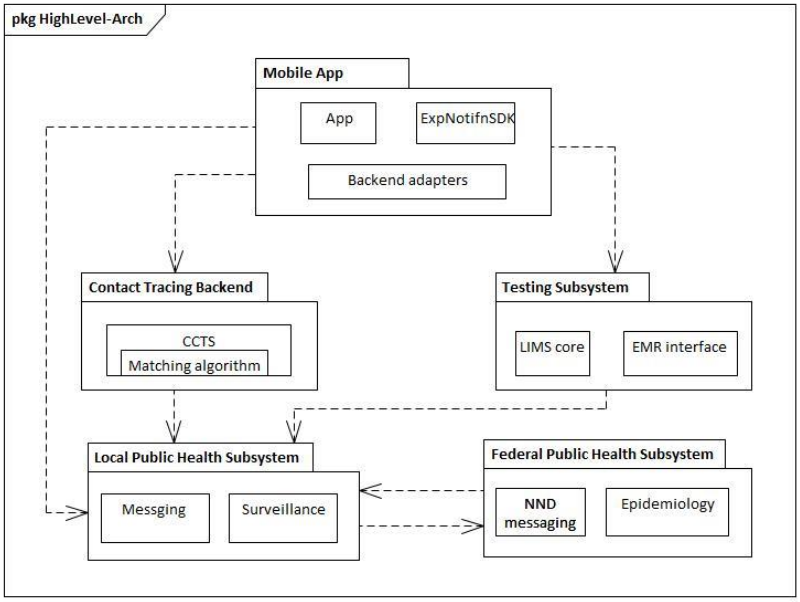

Figure 1 High level architecture : major components

Figure 1 shows the overall architecture with the major components involved in a contact tracing system. The app with its contact tracing exposure notification API and backend interface adapters constitute the software components on the smartphone. However, the crucial backend subsystem with a centralized contact tracing subsystem (CCTS), a laboratory information management system (LIMS) with its interface to some sort of electronic medical record system are critical as well. Finally, a robust public health surveillance system at the local and federal level with their messaging subsystems including nationally notifiable diseases (NND) and epidemiological capabilities are essential for the entire solution to work.

The major types of architecture in automated contact tracing are centralized and decentralized described below. The main difference is in the amount of work done and data stored by the central authority. The other important aspect is how identifiers are correlated and the intelligence in identification of a new infected case in the subsystem that runs this matching logic. One fundamental assumption in both the approaches is that there must be way to identify the smartphones that came to proximity and a way to link them to actual tests being done in testing facilities and laboratories.

A. Centralized design

The backend public health system stores the anonymous temporary IDs in this approach. Once someone tests positive, a matching algorithm is run centrally in a "Centralized Contact Tracing Subsystem" (CCTS) to identify the susceptible contact. A confirmation of the proximity proof of being close to an infected person and the duration/timestamp of that event are all that a digital contact tracing application needs, to alert the asymptomatic person. Other public health surveillance or EMR (electronic medical record) information systems with data about the infected person, their contacts, location of that contact, context for that contact etc. are unnecessary to be part of the contact tracing application on the phone. Figure 2 shows the typical sequence of messages in this centralized design.

1. CCTS is the source of the temporary identifiers that the smartphones use in their proximity protocol using Bluetooth LE technology. It generates and holds a long-term pseudo-identifier and generates the ephemeral pseudo-identities (EphIDs) for the smartphones.
2. These EphIDs are then received by smartphones that come to close proximity which locally stores them with the corresponding proximity and duration information. The UML sequence diagram in Figure 2 shows two people, Ryan with an iPhone and Eric with an Android based phone (i.e., Galaxy) come into contact [6]. Both the smartphones record the ephemeral identifiers including duration and proximity information.
3. When Ryan tests positive for the virus after showing symptoms, he provides his intent to share the results of the testing outcome to the local centralized contact tracing system (CCTS) via a public health application or his contact tracing application with such a backend integration on his iPhone. First, he receives an authentication code from the public health authority and then uploads his results. This step is important for sheer privacy reasons.
4. CCTS subsystem now runs a matching algorithm that determines a score for all the people that came to Ryan's proximity in a certain time window. Eric's Android phone eventually gets an alert from the authorities as he was identified as someone who came in close contact with Ryan.
5. The centralized system also updates the central public health federal system so that epidemiological research can be undertaken by the scientists at the federal level with this data.

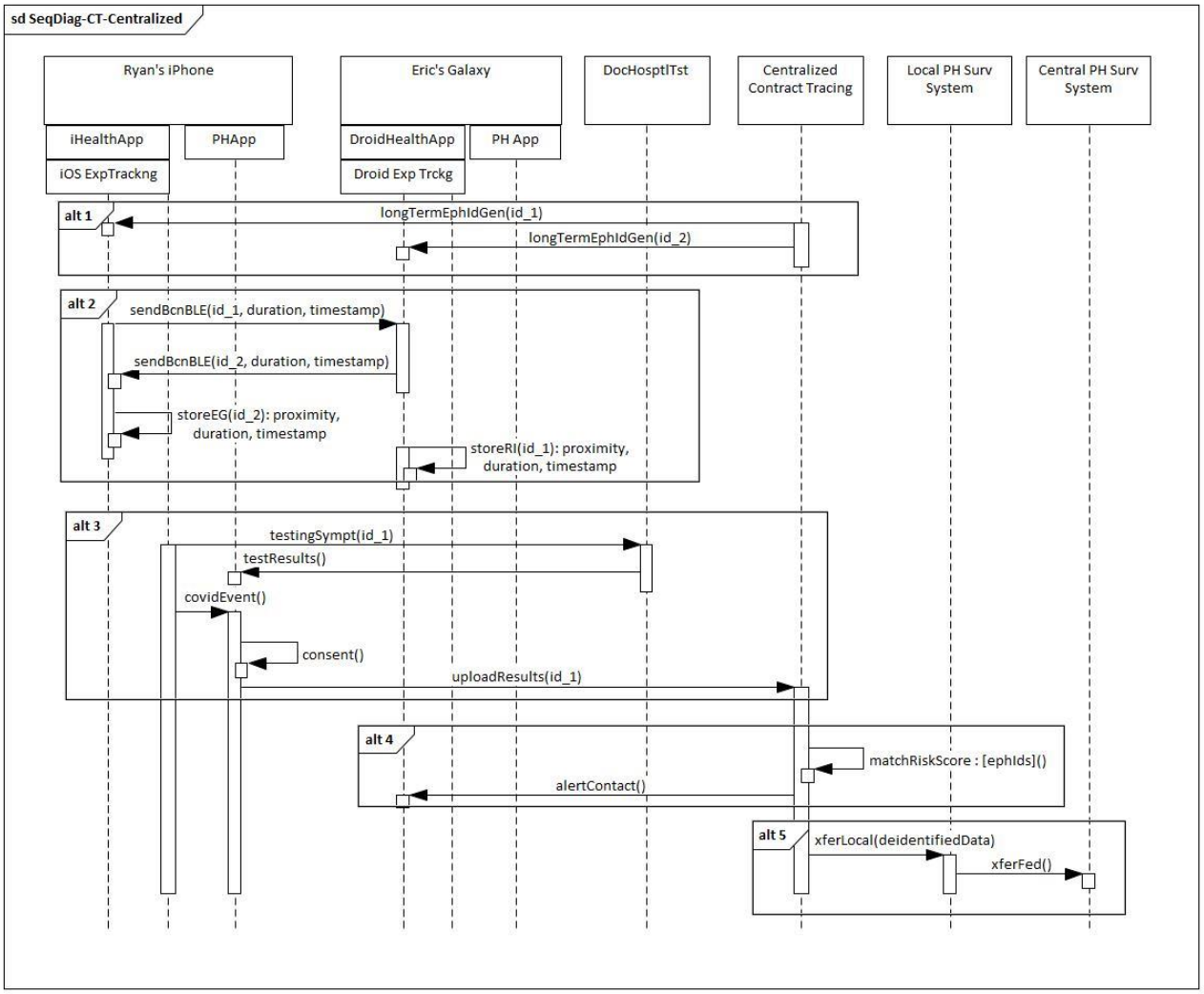

Figure 2 Sequence diagram : centralized approach

B. Decentralized Design

This approach keeps the minimal amount of data and logic as part of the core contact tracing backend subsystem. Identification of the underlying contact network graph involves minimal data about the infected person and its contacts. Figure 3 shows the typical sequence of messages in this centralized design. The "Local PH Surv System" stores both the anonymized IDs from the phones as well as reported positive cases. The correlation of the temporary IDs on the phone and the test result is a prerequisite to the creation of the list of "infected" people. This list is what is downloaded on the phone as shown in the activity in the process. The identifiers are good enough to run the matching algorithm locally on the smartphone.

1. Smartphones locally generate and broadcast via BlutoothLE technology frequently changing ephemeral identifiers (EphIDs). Neighboring smartphones listen to these EphIDs and store them together with the duration and timestamp.
2. Once Ryan tests positive and provides consent, his phone uploads test results to the local public health system.
3. At a certain interval, all smartphones download the list of infected patients who tested positive and reconstruct the corresponding EphIDs locally. If the smartphone has a record of any of these infected EphIDs, then the user must have been in contact with an infected person and the application computes the owner's risk score. If this score is above the threshold, the smartphone initiates a notification process.

Eric's Android phone periodically downloads the list of infected people within a time window (e.g., last 10 days) and can detect locally that he did come in contact with Ryan.

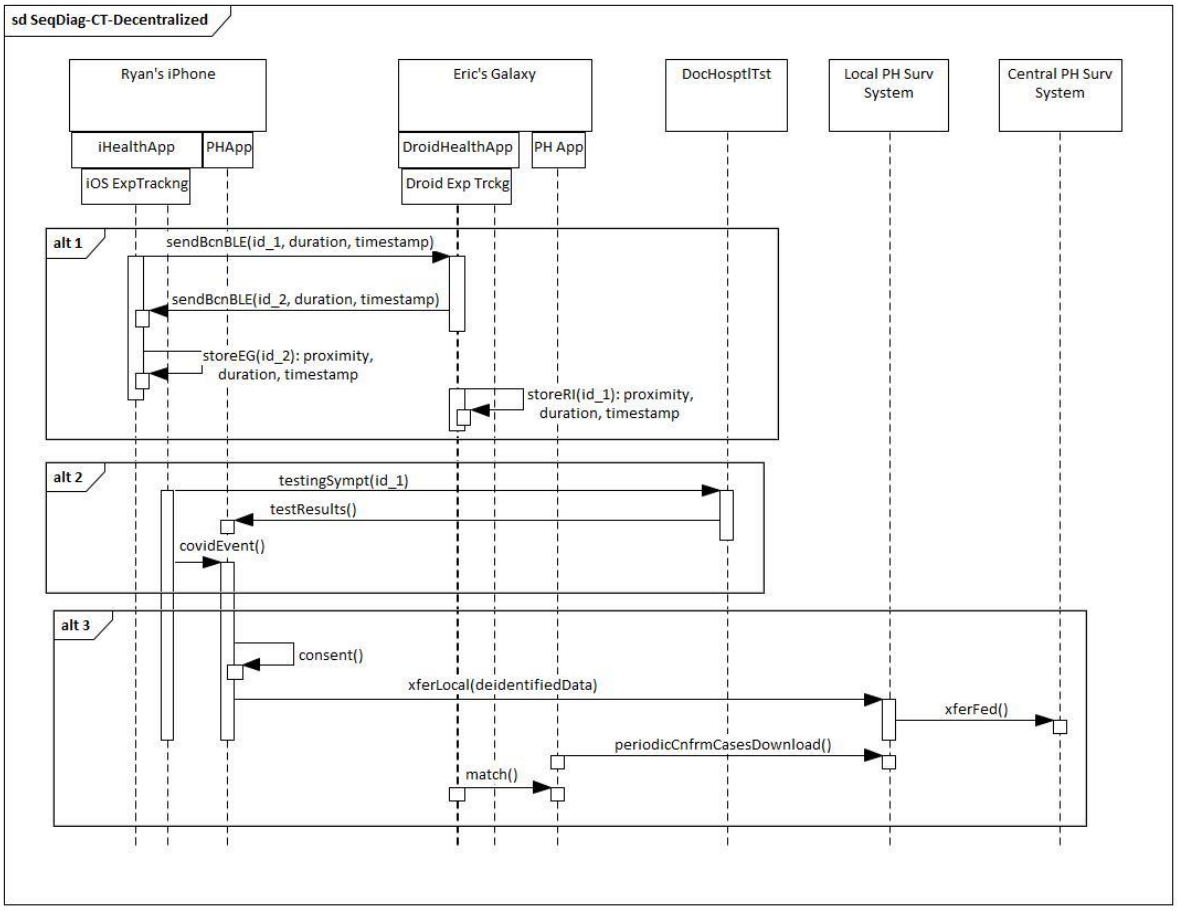

Figure 3 Sequence diagram : decentralized approach

C. Deployment view

The implementation of the designs outlined in the previous section has multiple options to use of any of the modern software development stacks. The main components as shown in Figure 4 are

1. Front-end app platforms based on iOS and Android : These can be a single integrated contact tracing app with a built-in public health component or can be two separate apps where the contact tracing app is dependent on the public health apps
2. Contact tracing APIs : The Exposure Notification APIs are typically the interfaces offered by the two leading providers Apple [6][7] and Google [8]. They are packaged as part of the SDKs (software development kits) of iOS and Android platforms.
3. Public health contact tracing service : This is CCTS for the centralized option with the matching algorithm that also provides the temporary identifiers for correlation to the ephemeral identifiers.
4. Local or state public health surveillance system : These are public health infrastructures at local, regional or states level that handles the data via messaging in a loosely coupled hierarchical public health network [9]. This is typically the level at which most of the actual epidemiological analysis and research take place.
5. Testing facilities or laboratories reporting system : These are laboratory facilities and test centers where actual testing takes place and the source of test data that gets delivered to the applications and the backend systems.

Deployment of the different components listed above can have multiple configurations. Figure 4 illustrates such a typical configuration. As an example, the centralized component can be hosted in an ‘on-prem’ data center as would be the case with many healthcare providers and the federal public health or disease reporting systems. On the other hand, laboratories and regional or state health systems can be on a public or private cloud. Any implementation needs to work out the cost benefits, project timelines, network and security issues before embarking on the actual development lifecycle. The red arrows show the information flow between components in the centralized design while the others are common to both.

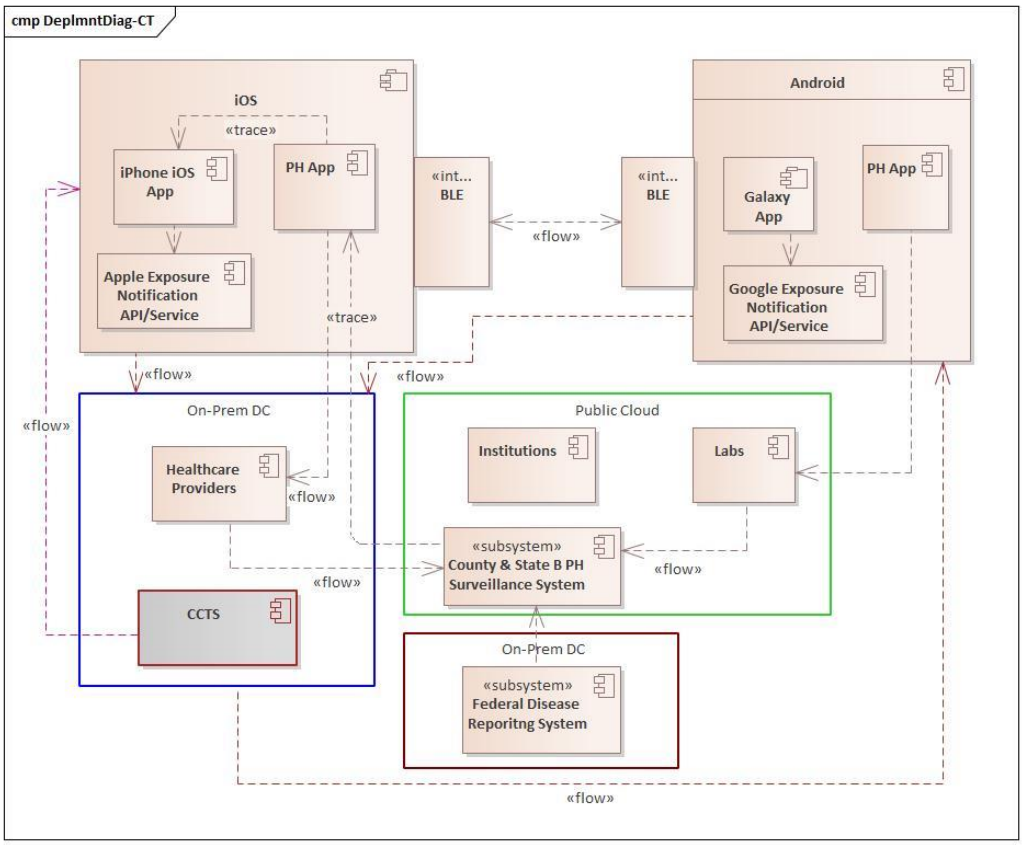

Figure 4 Deployment diagram of essential components

III. DISCUSSION

A. Security and Privacy Issues

Automated digital contact tracing as an aid to support of public health response is a top priority now. A lot of countries are trying to accelerate the use of these proposed apps and their associated backend components which are predominantly centralized by the Federal governments resulting in debates around privacy concerns [10]. At one end of the spectrum are countries like China, South Korea, Singapore with strong centralized governments and advanced technology and surveillance capabilities. At the other end of the spectrum are European countries like Germany or the United States who have the technological capability but there is an intense debate and politics involving federal and state governments on policies and privacy concerns. A good example is Germany where the decision to build a centralized architecture for contact tracing suffered a setback and the final decision converged on a decentralized architecture [11]. Concerns about allowing the Government access to pseudonymized proximity data and social graph of individuals in the society are raising alerts of state surveillance by privacy experts[12].

The Decentralized Privacy-Preserving Proximity Tracing (DP-3T) project is an open protocol for COVID-19 proximity tracing produced by a core team of over 25 scientists and academic researchers from across Europe [13]. It has also been scrutinized and improved by the wider community. DP-3T members have been participating in the

loose umbrella of the 'Pan-European Privacy-Preserving Proximity Tracing' (PEPP-PT) project [14]. DP-3T is not the only protocol under this umbrella. PEPP-PT also endorses centralized approaches with very different privacy properties. Pandemics do not respect borders, so there is substantial value in PEPP-PT's role of encouraging dialogue, knowledge-sharing, and interoperability.

An interesting example of centralized architecture is India's ArogyyaSetu [15] application that not only uses location services, stores PII such as Professional and other sensitive personal data but also is mandatory for citizens to download. The benefits of such solutions created most likely as a quick response by technology teams without much architectural thought or domain expertise are questionable in countries with a poor or almost non-existing public health surveillance infrastructure. Singapore, on the contrary, has created TraceTogether [16] app using the framework suggested by MIT that uses the centralized architecture described here without using location services of the smartphones.

IV. Conclusion

Before contact tracing applications are rolled out, there needs to be a cohesive policy and agreements between Central, Local governments and privacy advocacy groups. Informed leadership with collaboration between public health experts, epidemiologists and technology experts can create all-encompassing solutions in an expedient manner that can serve the communities. It is a great opportunity to engage and leverage the expertise and collaboration of Enterprise Architecture and Medical Informatics professionals to address the extraordinary challenges of the day. Unless a cohesive picture of the entire ecosystem is defined and appreciated, proper investment decisions cannot be made in a phased manner to identify and tackle the priorities. The objective of this paper is to show a comprehensive view from the standpoint of Enterprise Architecture of all the aspects involved in digital contact tracing and illustrate the point that an app in a phone, as described in popular media, is just the tip of the iceberg. The discourse needs to happen at the level described here to make serious progress to automate and advance the field of medical informatics.

References

[1] Neil Bailey, “The calculus of death shows the COVID lock-down is clearly worth the cost”, https://bit.ly/35X6HKL, The Conversation, 2020.
[2] Greg Stewart, Klaske van Heusden and Guy A. Dumont, “How Control Theory Can Help Us Control Covid-19” https://bit.ly/2XaUy11, IEEE Spectrum, April, 2020.
[3] Neil M. Ferguson, “Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand”, March 16, 2020.
[4] Cameron Nowzari, Victor M. Preciado and Geirge J. Pappas, “Analysis and Control of Epidemics : A Survery of Spreading Process on Complex Networks”, IEEE Control Systems Magazine, February, 2016.
[5] Mary Shaw and David Garlan, “Software Architecture : Perspectives on an Emerging Discipline”, Pearson, April, 1996.
[6] Juli Clover, “Apple's Exposure Notification System: Everything You Need to Know”, https://bit.ly/3bKMk4V, MacRumors, May 4, 2020.
[7] Dev “Building an App to Notify Users of COVID-19 Exposure” https://apple.co/3dCVL7J, April, 2020.
[8] Dev Team{Exposure Notification : Android API Documentation}, v1.2, April, 2020
[9] Dibyendu Baksi, “Formal interaction specification in public health surveillance systems using π-calculus”, Computer Methods and Programs in Biomedicine, vol 92, pp 115-120, 2008.
[10] COVID-19 Apps, Wikipedia, https://en.wikipedia.org/wiki/COVID-19_apps, May 2020.
[11] Natasha Lomas, “Germany ditches centralized approach to aoo for COVID-19 contacts tracing”, https://tcrn.ch/3ct1QU5, April, 2020.
[12] Zak Doffman, “Forget Apple And Google—Contact-Tracing Apps Just Dealt Serious New Blow”, https://bit.ly/2X3e7s7, May, 2020.
[13] Carmela Troncoso et al, “Decentralized Privacy-Preserving Proximity Tracing”, https://github.com/DP-3T/documents , github, April 2020.
[14] PEPP-PT, “Pan-European Privacy-Preserving Proximity Tracing”, https://www.pepp-pt.org/ , March 2020.
[15] Anand Venkatanarayanan, “Covid-19 : How The Aarogya Setu App Handles Your Data”, https://bit.ly/35TSE8W, BloombergQuintOpinion, April, 2020.
[16] Govt of Singapore, TraceTogether, https://www.tracetogether.gov.sg/, May 2020.