Chapter 1

Introduction

Contact tracing is a non-pharmaceutical intervention that aims to halt the spread of infectious disease by identifying and quarantining individuals that have been in close physical proximity with the infected [7, 37]. Historically, contact tracing has been conducted manually

In response to the pandemic of coronavirus disease 2019 (COVID-19) [17, 42, 48], numerous approaches of *automatic contact tracing* (ACT) have been proposed [39, 41].

Shubina et al. [41] provide an in-depth survey while Reichert, Brack, and Scheuermann [39] specifically focuses on techniques that utilize Bluetooth Low Energy (BLE) and provides a more current analysis of message-based approaches.

Effective contact tracing is inherently difficult because of the complex interactions between epidemiology, public health, ethics, politics, and sociocultural norms [7].

Privacy risk of message-based approaches is that an adversary can monitor network traffic in order to reconstruct the contact network. Using a decentralized data layer would mitigate this attack mode.

Higher-order contact tracing has demonstrated increased efficacy [37]

The remainder of this work is organized as follows.

Chapter 2

Risk Propagation

Risk propagation is a message-passing algorithm that estimates an individual's infection risk by considering their demographics, symptoms, diagnosis, and contact with others. Formally, a *risk score* $s_t \in [0, 1]$ is a timestamped infection probability where $t \in \mathbb{N}$ is the time of its computation. Thus, an individual with a high risk score is likely to test positive for the infection and poses a significant health risk to others. There are two types of risk scores: *symptom scores*, or prior infection probabilities, which account for an individual's demographics, symptoms, and diagnosis [8, 32]; and *exposure scores*, or posterior infection probabilities, which incorporate the risk of direct and indirect contact with others.

Given their recent risk scores and contacts, an individual's exposure score is derived by marginalizing over the joint infection probability distribution. Naively computing this marginalization scales exponentially with the number of variables (i.e., individuals). To circumvent this intractability, the joint distribution is modeled as a factor graph, and an efficient message-passing procedure is employed to compute the marginal probabilities with a time complexity that scales linearly in the number of factor vertices (i.e., contacts).

Let $G = (X, F, E)$ be a *factor graph* where X is the set of variable vertices, F is the set of factor vertices, and E is the set of edges incident between them [26]. A *variable vertex* $x : \Omega \to \{0, 1\}$ is a random variable that represents the infection status of an individual, where the sample space is $\Omega = \{\textit{healthy}, \textit{infected}\}$ and

$$x(\omega) = \begin{cases} 0 & \text{if } \omega = \textit{healthy} \\ 1 & \text{if } \omega = \textit{infected}. \end{cases}$$

Thus, $p_t(x_i) = s_t$ is a risk score of the i-th individual. A *factor vertex* $f : X \times X \to [0, 1]$ defines the transmission of infection risk between two contacts. Specifically, contact between the i-th and j-th individual is represented by the factor vertex $f(x_i, x_j) = f_{ij}$, which is adjacent to the variable vertices x_i, x_j. Figure 2.1 depicts a factor graph that reflects the domain constraints.

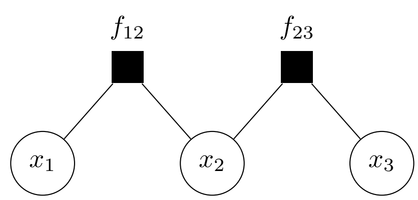

Figure 2.1: A factor graph of 3 variable vertices and 2 factor vertices.

2.1 Synchronous Risk Propagation

Ayday, Yoo, and Halimi [2] first proposed risk propagation as a synchronous, iterative message-passing algorithm that uses the factor graph to compute exposure scores. The first input to Risk-Propagation is the set family S, where

$$S_i = \{\, s_t \mid \tau - t < T_s \,\} \in S \tag{2.1}$$

is the set of recent risk scores of the i-th individual. The second input to Risk-Propagation is the contact set

$$C = \{\, (i, j, t) \mid i \neq j, \tau - t < T_c \,\} \tag{2.2}$$

such that (i, j, t) is the *most recent* contact between the i-th and j-th individual that occurred from time t until at least time $t + \delta$, where $\delta \in \mathbb{N}$ is the *minimum contact duration*. Naturally, risk scores and contacts have finite relevance, so (2.1) and (2.2) are constrained by the *risk score expiry* $T_s \in \mathbb{N}$ and the *contact expiry* $T_c \in \mathbb{N}$, respectively. The *reference time* $\tau \in \mathbb{N}$ defines the relevance of the inputs and is assumed to be the time at which Risk-Propagation is invoked. For notational simplicity in Risk-Propagation, let X be a set. Then $\max X = 0$ if $X = \emptyset$.

2.1.1 Variable Messages

The current exposure score of the i-th individual is defined as $\max S_i$. Hence, a *variable message* $\mu_{ij}^{(n)}$ from the variable vertex x_i to the factor vertex f_{ij} during the n-th iteration is the set of maximal risk scores $R_i^{(n-1)}$ from the previous $n-1$ iterations that were not derived by f_{ij}. In this way, risk propagation is reminiscent of the max-sum algorithm; however, risk propagation aims to maximize *individual* marginal probabilities rather than the joint distribution [6, pp. 411–415].

2.1.2 Factor Messages

A *factor message* $\lambda_{ij}^{(n)}$ from the factor vertex f_{ij} to the variable vertex x_j during the n-th iteration is an exposure score of the j-th individual that is based on interacting with those at most $n-1$ degrees separated from the i-th individual. This population is defined by the subgraph induced in G by

$$\{ v \in X \cap F \setminus \{x_j, f_{ij}\} \mid d(x_i, v) \leq 2(n-1) \},$$

where $d(u, v)$ is the shortest-path distance between the vertices u, v. The computation of a factor message assumes the following.

1. Contacts have a nondecreasing effect on an individual's exposure score.

2. A risk score s_t is *relevant* to the contact (i, j, t_{ij}) if $t < t_{ij} + \beta$, where $\beta \in \mathbb{N}$ is a *time buffer* that accounts for the incubation period, along with the delayed reporting of symptom scores and contacts. The time buffer is also known as the *(contact) tracing window* [38] and is similar to the *notification window* used by other contact tracing applications [28].

3. Risk transmission between contacts is incomplete. Thus, a risk score

decays exponentially along its transmission path in G at a rate of $\log \alpha$, where $\alpha \in (0, 1)$ is the *transmission rate*. Figure 2.2 visualizes this decay, assuming a transmission rate of $\alpha = 0.8$ [19].

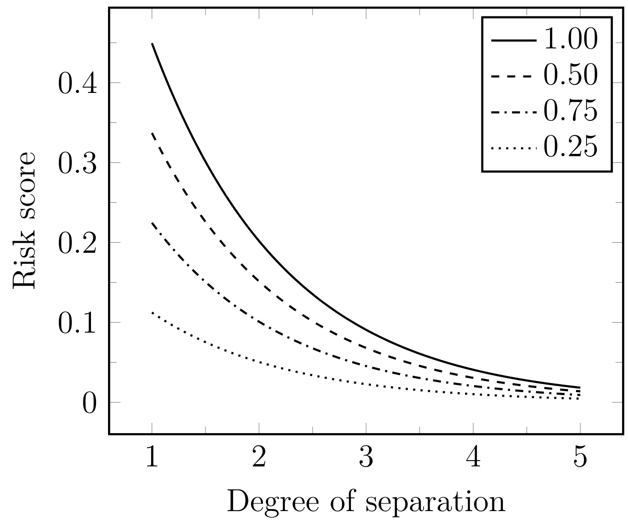

Figure 2.2: Exponential decay of risk scores.

To reiterate, a factor message $\lambda_{ij}^{(n)}$ is the maximum relevant risk score in the variable message $\mu_{ij}^{(n)}$ (or 0) that is scaled by the transmission rate α.

Ayday, Yoo, and Halimi [2] assume that the contact set C may contain (1) multiple contacts between the same two individuals and (2) invalid contacts, or those lasting less than δ time. However, these assumptions introduce unnecessary complexity. Regarding assumption 1, suppose the i-th and j-th individual come into contact m times such that $t_k < t_\ell$ for $1 \leq k < \ell \leq m$. Let Λ_k be the set of relevant risk scores, according to the contact time t_k, where

$$\Lambda_k = \{ \alpha s_t \mid s_t \in \mu_{ij}^{(n)}, t < t_k + \beta \} .$$

Then $\Lambda_k \subseteq \Lambda_\ell$ if and only if $\max \Lambda_k \leq \max \Lambda_\ell$. Therefore, only the most recent contact time t_m is required to compute the factor message $\lambda_{ij}^{(n)}$. With respect to assumption 2, there are two possibilities.

1. If an individual has at least one valid contact, then their exposure score is computed over the subgraph induced in G by their contacts that define the neighborhood N_i of the variable vertex x_i.

2. If an individual has no valid contacts, then their exposure score is $\max S_i$ or 0, if all of their previously computed risk scores have expired.

In either case, a set C containing only valid contacts implies fewer factor vertices and edges in the factor graph G. Consequently, the complexity of RISK-PROPAGATION is reduced by a constant factor since fewer messages must be computed.

2.1.3 Termination

To detect convergence, the normed difference between the current and previous exposure scores is compared to the threshold $\epsilon \in \mathbb{R}$. Note that $\mathbf{r}^{(n)}$ is the vector of exposure scores in the the n-th iteration such that $r_i^{(n)}$ is the i-th component of $\mathbf{r}^{(n)}$. The L_1 and L_∞ norms are sensible choices for detecting convergence. Ayday, Yoo, and Halimi [2] use the L_1 norm, which ensures that an individual's exposure score changed by at most ϵ after the penultimate iteration.

RISK-PROPAGATION(S, C)

$(X, F, E) \leftarrow$ FACTOR-GRAPH(C)
$n \leftarrow 1$
for each $x_i \in X$
  $R_i^{(n-1)} \leftarrow$ top k of S_i
  $r_i^{(n-1)} \leftarrow \max R_i^{(n-1)}$
  $r_i^{(n)} \leftarrow \infty$
while $\|\mathbf{r}^{(n)} - \mathbf{r}^{(n-1)}\| > \epsilon$
  for each $\{x_i, f_{ij}\} \in E$
    $\mu_{ij}^{(n)} \leftarrow R_i^{(n-1)} \setminus \{\, \lambda_{ji}^{(\ell)} \mid \ell \in [1 \,..\, n-1] \,\}$
  for each $\{x_i, f_{ij}\} \in E$
    $\lambda_{ij}^{(n)} \leftarrow \max \{\, \alpha s_t \mid s_t \in \mu_{ij}^{(n)}, t < t_{ij} + \beta \,\}$
  for each $x_i \in X$
    $R_i^{(n)} \leftarrow$ top k of $\{\, \lambda_{ji}^{(n)} \mid f_{ij} \in N_i \,\}$
  for each $x_i \in X$
    $r_i^{(n-1)} \leftarrow r_i^{(n)}$
    $r_i^{(n)} \leftarrow \max R_i^{(n)}$
  $n \leftarrow n + 1$
return $\mathbf{r}^{(n)}$

2.2 Asynchronous Risk Propagation

While RISK-PROPAGATION offers proof of concept, it is not viable for real-world application. RISK-PROPAGATION is an *offline algorithm* [11], because it requires the contact and health information of all individuals as input. As

Ayday, Yoo, and Halimi [2] note, this centralization of personal data is not privacy-preserving. Risk-Propagation also introduces communication overhead and computational redundancy since most exposure scores are unlikely to change across frequent invocations. To mitigate this inefficiency, Ayday et al. [3] suggest running Risk-Propagation once or twice per day. Unfortunately, this cadence incurs substantial delay in updating individuals' exposure scores. In the midst of a pandemic, timely information is essential for individual and collective health. To address the limitations of Risk-Propagation, Ayday, Yoo, and Halimi [2] propose decentralizing the factor graph such that the processing entity associated with the i-th individual maintains the state of the i-th variable vertex and the neighboring factor vertices. Applying one-mode projection onto the variable vertices [47], Figure 2.3 illustrates how each entity corresponds to a portion of the factor graph. More generally, Ayday, Yoo, and Halimi [2] envision risk propagation as a decentralized communication protocol for informing individuals about their infection risk. Such a message-passing protocol naturally aligns with the *actor model*, a local model of concurrent computing that defines computation as patterns of message passing amongst computational entities called *actors* [1, 10, 14, 20, 21, 22]. Since communication is asynchronous in the actor model, risk propagation defined in this way is called *asynchronous risk propagation*.

2.2.1 Actor Behavior

An actor in the ShareTrace actor system corresponds to an individual. The Create-Actor operation [1] initializes an actor a with the following at-

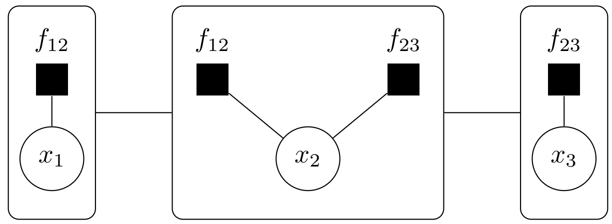

Figure 2.3: One-mode projection onto the variable vertices in Figure 2.1.

tributes.

- *a.exposure*: the current exposure score of the individual. This attribute is either a symptom score, a risk score sent by another actor, or the null risk score (see Null-Risk-Score).

- *a.contacts*: a *dictionary* (see **??**) of contacts. In the context of an actor, a contact is a *proxy* [15] of the actor that represents an individual with which the individual represented by this actor was physically proximal. That is, if the i-th individual interacted with the j-th individual, then a_i*.contacts* contains a contact c such that *c.key* = *c.name* is a name of the j-th actor and *c.time* is the most recent time of contact. This attribute extends the concept of *actor acquaintances* [1, 20, 21] to be time-varying.

- *a.scores*: a dictionary of exposure scores such that *s.key* for an exposure score s is the time interval during which *a.exposure* = s. The null risk score is returned for queries in which the dictionary does not contain a risk score with a key that intersects the given query interval. Figure 2.4 depicts a hypothetical step function that *a.scores* represents.

Null-Risk-Score

$s.value \leftarrow 0$

$s.time \leftarrow 0$

return s

Create-Actor

$a.contacts \leftarrow \emptyset$

$a.scores \leftarrow \emptyset$

$a.exposure \leftarrow$ Null-Risk-Score

return a

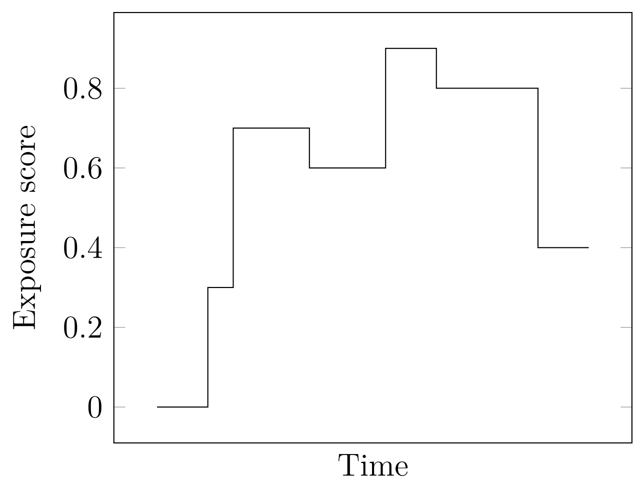

Figure 2.4: Hypothetical history of an individual's exposure score.

The interface of an actor is primarily defined by two types of messages: contacts and risk scores. Based on Section 2.1, let the *time to live* (TTL) of a message be the remaining time of its relevance. The reference time τ is assumed to be the current time.

```
RISK-SCORE-TTL(s)
  return T_s − (τ − s.time)
```

```
CONTACT-TTL(c)
  return T_c − (τ − c.time)
```

The HANDLE-RISK-SCORE operation defines how an actor behaves upon receiving a risk score. The key *s.key* initially associated with the risk score s is the time interval during which it is relevant. For the dictionary *a.scores*, MERGE preserves the mapping invariant defined above such that risk scores are ordered first by value and then by time. Thus, $s.key \subseteq [s.time, s.time+T_s)$ for all exposure scores in *a.scores*. The UPDATE-EXPOSURE-SCORE operation is the online equivalent of updating an individual's exposure score in RISK-PROPAGATION.

```
HANDLE-RISK-SCORE(a, s)
  if RISK-SCORE-TTL(s) > 0
      s.key ← [s.time, s.time + T_s)
      MERGE(a.scores, s)
      UPDATE-EXPOSURE-SCORE(a, s)
      for each c ∈ a.contacts
          APPLY-RISK-SCORE(a, c, s)
```

Update-Exposure-Score(a, s)

if $a.exposure.value < s.value$

  $a.exposure \leftarrow s$

else if Risk-Score-Ttl($a.exposure$) ≤ 0

  $a.exposure \leftarrow$ Maximum($a.scores$)

For the moment, assume that Apply-Risk-Score is the online equivalent of computing a factor message (see Section 2.1). Line 2 indicates that a copy s' of the risk score s is updated using the transmission rate α. The Send operation enables actor communication [1].

Apply-Risk-Score(a, c, s)

1: **if** $c.time + \beta > s.time$

2:   $s'.value \leftarrow \alpha \cdot s.value$

3:   Send($c.name, s'$)

The problem with Apply-Risk-Score is that it causes risk scores to propagate *ad infinitum*. Unlike Risk-Propagation, a global convergence test is not available to terminate message passing, so it is necessary to define a local condition that determines if a risk score should be sent to another actor. Practically, "self-terminating" message-passing is necessary for asynchronous risk propagation to be scalable and cost-efficient.

The intent of sending a risk score to an actor is to update its exposure score. According to Handle-Risk-Score, it is only necessary to send an actor risk scores with values that exceed its current exposure score. Thus, an actor can associate a *send threshold* with a contact such that the target actor only receives risk scores that exceed the threshold. To permit a trade-off between accuracy and efficiency, let the *send coefficient* $\gamma \in \mathbb{R}$ and *tolerance* $\epsilon \in \mathbb{R}$ be scaling factors that are applied to a risk score upon setting the send threshold.

SET-SEND-THRESHOLD(c, s)

1: $s'.value \leftarrow \gamma \cdot s.value + \epsilon$
2: $c.threshold \leftarrow s'$

The APPLY-RISK-SCORE operation that incorporates the concept of a send threshold is defined below. Assuming a finite number of actors, a positive send coefficient guarantees that a risk score has finite propagation.

APPLY-RISK-SCORE(a, c, s)

1: **if** $c.threshold.value < s.value$ **and** $c.time + \beta > s.time$
2: $\quad s'.value \leftarrow \alpha \cdot s.value$
3: $\quad$ SET-SEND-THRESHOLD(c, s')
4: $\quad$ SEND$(c.name, s')$

The SET-SEND-THRESHOLD operation defines *how* the send threshold is updated, but not *when* it should be updated; the UPDATE-SEND-THRESHOLD operation encapsulates the latter. The second predicate on Line 1 stems from the fact that the send threshold is a risk score and thus subject to expiry. The first predicate, however, is more subtle and will be revisited shortly. The MAXIMUM-OLDER-THAN operation is the same as MAXIMUM, with the constraint that the key of the risk score intersects the interval $(-\infty, c.time + \beta)$. Thus, the returned risk score is always relevant to the contact. Consistent with Apply-Risk-Score, the risk score retrieved from $a.scores$ is scaled by the transmission rate and set as the new send threshold.

Update-Send-Threshold(a, c)

1: **if** $c.threshold.value > 0$ **and** Risk-Score-Ttl$(c.threshold) \leq 0$
2: $\quad s \leftarrow$ Maximum-Older-Than$(a.scores, c.time + \beta)$
3: $\quad s'.value \leftarrow \alpha \cdot s.value$
4: $\quad$Set-Send-Threshold(c, s')

The send threshold should be current when evaluating Line 1 of Apply-Risk-Score above, so Update-Send-Threshold is invoked beforehand.

Apply-Risk-Score(a, c, s)

1: Update-Send-Threshold(a, c)
2: **if** $c.threshold.value < s.value$ **and** $c.time + \beta > s.time$
3: $\quad s'.value \leftarrow \alpha \cdot s.value$
4: $\quad$Set-Send-Threshold(c, s')
5: $\quad$Send$(c.name, s')$

Returning to the first predicate on Line 1 of Update-Send-Threshold, the send threshold has a value of 0 when initially setting the send threshold as the null risk score and when no key in $a.scores$ intersects the query interval on (Line 2). Suppose the first predicate is omitted from Line 1. Given that Update-Send-Threshold is the first statement in Apply-Risk-Score, it is possible that the send threshold is set prior to sending the target actor a relevant risk score. In the worst case, this prevents *all* risk scores from being sent to that actor. As a result, that individual would not receive risk scores that may impact their exposure score. Therefore, for correct message-passing behavior, the send threshold is updated only when its value is nonzero.

The aforementioned refinements to Apply-Risk-Score have focused on ensuring that message passing terminates and behaves correctly over time. To conclude the definition of Apply-Risk-Score, a message-passing optimization, called *sender-side aggregation* [31], is incorporated. Over a given period of time, an actor may receive several risk scores that are subsequently sent to multiple target actors. Rather than sending multiple risk scores, it would be more efficient to just send the final risk score. As a heuristic, Apply-Risk-Score can be modified so that a contact "buffers" a risk score that the target actor should receive.

```
Apply-Risk-Score(a, c, s)
Update-Send-Threshold(a, c)
if c.threshold.value < s.value and c.time + β > s.time
    s′.value ← α · s.value
    Set-Send-Threshold(c, s′)
    c.buffered ← s′
```

When the actor receives a periodic *flush timeout* message, all contacts are "flushed" by sending the buffered risk scores to the target actors. For convenience, the Handle-Flush-Timeout operation also removes all expired contacts from $a.contacts$.

```
Handle-Flush-Timeout(a)
for each c ∈ a.contacts
    if c.buffered ≠ NIL
        Send(c.name, c.buffered)
        c.buffered ← NIL
    if Contact-Ttl(c) ≤ 0
        Delete(a.contacts, c)
```

The Handle-Contact operation concludes this section on actor behavior. Similar to Handle-Risk-Score, expired contacts are not processed. The Merge operation for *a.contacts* differs from its usage with *a.scores*. That is, if a contact with the same key already exists, its contact time is updated to that of the newer contact; all other state of the previous contact is maintained. A risk score from *a.scores* is also applied to the contact to ensure that, if the actor receives no other risk score before the contact expires, the target actor is sent at least one relevant risk score.

```
Handle-Contact(a, c)
if Contact-Ttl(c) > 0
    c.threshold ← Null-Risk-Score
    c.buffered ← NIL
    c.key ← c.name
    Merge(a.contacts, c)
    s ← Maximum-Older-Than(a.scores, c.time + β)
    Apply-Risk-Score(a, c, s)
```

2.2.2 Message Reachability

The ShareTrace actor system is a *contact network*, a type of *temporal network* in which vertices represent individuals and edges indicate contact between them [23, 24]. The contact set defined in (2.2) is the *contact sequence* representation of a contact network [24]. Contact networks are typically used in epidemiological studies that aim to model and analyze the spreading dynamics of infection [12, 13, 25, 29, 35, 40, 49]. The ShareTrace actor network, however, models the spreading of infection *risk*. Holme and Saramäki [24] note that the transmission graph proposed by Riolo, Koopman, and Chick [40] "cannot handle edges where one node manages to not catch the disease." By framing the spreading phenomenon as continuous, rather than discrete, it is possible to model partial transmission.

A primitive of temporal reachability analysis is the *time-respecting path*: a contiguous sequence of contacts with nondecreasing time. Thus, vertex v is *temporally reachable* from vertex u if there exists a time-respecting path from u to v [33]. The following derivatives of a time-respecting path help quantify reachability in temporal networks [24].

- *influence set* $I(v)$: vertices that v can reach by a time-respecting path.
- *source set* $S(v)$: vertices that can reach v by a time-respecting path.
- *reachability ratio* $\rho(G)$: the average influence set cardinality in G.

Generally, a message-passing algorithm specifies constraints that determine when and what messages are sent between vertices. Even if operating on a temporal network, those constraints may be more or less strict than requiring temporal reachability. As a dynamic process, message passing on a time-varying network requires a broader definition of reachability that accounts for network topology *and* message-passing semantics [4].

Formally, the *reachability* of a message m from vertex u to vertex v is the number of edges along the shortest path P that satisfy the message-passing constraints,

$$r(u, v) = \sum_{(i,j) \in P} f(u, i, j, v),$$

where

$$f(u, i, j, v) = \begin{cases} 1 & \text{if all constraints are satisfied} \\ 0 & \text{otherwise.} \end{cases}$$

Vertex v is *reachable* from vertex u for the message m if there exists a shortest path P such that $r(u, v) = |P|$. The *reachability* of a message m from vertex u is

$$r(u) = \max_{v \in V} r(u, v). \tag{2.3}$$

Measures of temporal reachability can be extended to message reachability,

$$I(u) = \{\, v \in V \mid r(u, v) = |P| \,\}$$

$$S(v) = \{\, u \in V \mid r(u, v) = |P| \,\}$$

$$\rho(G, M) = \sum_{v \in V,\, m \in M} |I(v)| \cdot |V|^{-1},$$

where M is the set of messages associated with the vertices V.

Asynchronous Risk Propagation

Let P be the set of contact edges along the shortest path from actor u to actor v such that the actors are enumerated $1, \ldots, |P|$. Let (s_u, t_u) be a symptom score of actor u. Let s_{ij} be the value of the risk score that is associated with the send threshold of the i-th actor for the j-th actor. Finally, let t_{ij} be the most recent contact time between the the i-th and j-th individual. Then message reachability for asynchronous risk propagation is defined as

$$r(u, v) = \sum_{(i,j) \in P} [\alpha^i s_u > \gamma \alpha s_{ij} + \epsilon] \cdot [t_u < t_{ij} + \beta], \tag{2.4}$$

where $[\cdot]$ is the Iverson bracket[1].

By relaxing the temporal constraint in (2.4), an upper bound on the reachability of a symptom score can be defined. Reversing the inequality of the first term in (2.4) and solving for i,

$$\hat{r}(u, v) \leq \begin{cases} 0 & \text{if } s_u = 0 \\ |P| & \text{if } s_v = 0 \\ \log_\alpha \dfrac{\gamma \alpha s_v + \epsilon}{s_u} & \text{otherwise,} \end{cases} \tag{2.5}$$

where $\gamma \alpha s_v + \epsilon$ is the send threshold of the actor that precedes actor v along the path P. Assuming a transmission rate of $\alpha = 0.8$, a send coefficient of $\gamma = 1$, and a tolerance of $\epsilon = 0$, Figure 2.5 visualizes (2.5).

[1]The *Iverson bracket* $[x]$ is the indicator function of the set values for which x is true.

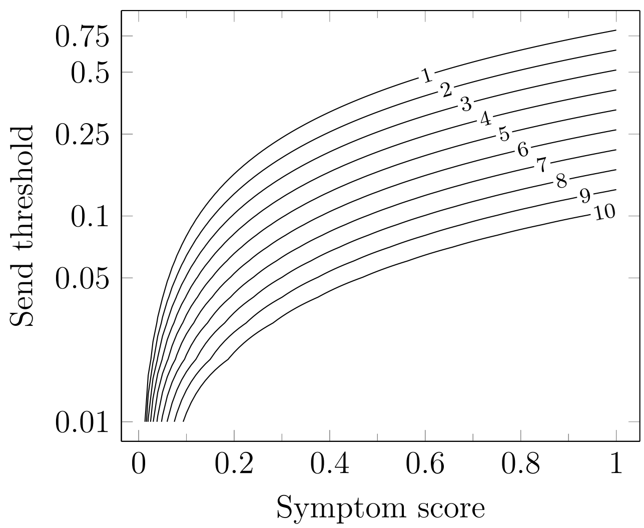

Figure 2.5: Message reachability for asynchronous risk propagation. Contour lines are shown for integral reachability values, indicating the maximum send threshold that permits sending the risk score.

2.3 Application Perspectives

With the algorithmic foundation of ShareTrace established, it is now possible discuss the aspects of system design and implementation. To a varying extent, prior work on ShareTrace [2, 3, **??**] has explored the practical considerations for deployment. This work, however, provides additional detail by contextualizing ShareTrace as an example of *mobile crowdsensing* (MCS), a "sensing paradigm that empowers ordinary citizens to contribute data sensed or generated from their mobile devices" that is aggregated "for crowd intelligence extraction and human-centric service delivery" [18]. To offer a clear comparison with other MCS applications, this section applies the classification criteria proposed by Capponi et al. [9]. Note, this section assumes the usage of asynchronous risk propagation (Section 2.2).

2.3.1 Application Layer

Task Scheduling *Proactive scheduling* allows users to decide when and where they contribute sensing data, while *reactive scheduling* requires that "a user receives a request, and upon acceptance, accomplishes a task" [9]. ShareTrace follows proactive scheduling where the sensing task is to detect proximal interactions with other individuals. Naturally, the scheduling of this task is at the discretion of ShareTrace users.

Task Assignment *Centralized assignment* assumes that "a central authority distributes tasks among users." Conversely, with *decentralized assignment*, "each participant becomes an authority and can either perform the task or forward it to other users" [9]. The latter aligns with ShareTrace since each user is responsible for their own interactions.

Task Execution With *single-task execution*, MCS applications assign one type of task to users, while *multi-task execution* assigns multiple types of tasks [9]. ShareTrace only involves the single task of sensing proximal interactions.

User Recruitment *Volunteer-based recruitment* is when citizens can "join a sensing campaign for personal interests…or willingness to help the community," while *incentive-based recruitment* promotes participation and offers control over the recruitment rate…These strategies are not mutually exclusive and users can first volunteer and then be rewarded for quality execution of sensing tasks" [9]. ShareTrace follows volunteer-based recruitment, though incentive mechanisms would be an important consideration for widespread adoption.

User Selection *User-centric selection* is when "contributions depend only on participants['] willingness to sense and report data to the central collector, which is not responsible for choosing them." *Platform-centric selection* is "when the central authority directly decides data contributors…Platform centric decisions are taken according to the utility of data to accomplish the targets of the campaign" [9]. ShareTrace employs user-centric selection, because the purpose of the application is to passively sense in-person interactions and provide individuals with knowledge of their infection risk.

User Type A *contributor* "reports data to the MCS platform with no interest in the results of the sensing campaign" and "are typically driven by incentives or by the desire to help the scientific or civil communities." A *consumer* joins "a service to obtain information about certain application scenario[s] and have a direct interest in the results of the sensing campaign" [9]. ShareTrace users could be consumers or contributors but would likelier be the latter.

2.3.2 Data Layer

Data Storage *Centralized storage* involves data being "stored and maintained in a single location, which is usually a database made available in the cloud. This approach is typically employed when significant processing or data aggregation is required." *Distributed storage* "is typically employed for delay-tolerant applications, i.e., when users are allowed to deliver data with a delayed reporting" [9].

ShareTrace relies on decentralized storage

The ShareTrace system architecture is purposely not built on the assumption of a personal data architecture [34], primarily due to there historical intractability in the consumer market. It is impractical to assume that all services and applications would utilize a single data model for an individual's personal data. Moreover, local storage of personal data is inherently more secure. The argument that personal data stores allow an individual to control the usage of their personal data is simply not valid. Technology alone cannot prevent the misuse of personal data. Lastly, the usage of remote, cloud-based computation introduces unnecessary complexity.

Approach 1: Secure pub-sub with IPFS

Approach 2: point-to-point messaging with DIDComm

The InterPlanetary FileSystem (IPFS)[2] [5, 44]

- Discussion around decentralization [45]
- Secure publish-subscribe
- Technologies: Trust Over IP, DIDComm[3]

The system model is similar to that in previous work [2, 3].

- Each user owns a *personal data store* (PDS), a form of cloud storage that empowers the user with ownership and access control over their data.
- Symptom scores are computed in a user's PDS to support integrating multiple streams of personal data [3]. While local symptom-score com-

[2] `https://ipfs.tech`

[3] `https://identity.foundation/didcomm-messaging/spec`

putation [2, 3] is more privacy-preserving, it is assumed that the user's PDS is a trusted entity.

- User device interactions serve as a proxy for proximal human interactions. This work does not assume a specific protocol, but does assume that the protocol can approximate the duration of contact with relative accuracy and that communication with the actors of those contacted users can be established in a privacy-preserving manner.

- No geolocation data is collected [3]. As a decentralized, proximity-based solution, it is not necessary to collect user geolocation data. See **??** for a discussion of a geolocation-based design that was considered.

Data Format *Structured data* is standardized and readily analyzable. *Unstructured data*, however, requires significant processing before it can be used [9]. ShareTrace data, such as reported symptoms, biometric data, contact identifiers, and risk scores, is structured.

Data Dimensionality *Single-dimension data* typically occurs when a single sensor is used, while *multi-dimensional data* arises with the use of multiple sensors. Prior work on ShareTrace assumes single-dimensional data: contact identifiers received via Bluetooth Low Energy (BLE). Though, the integration with biometric sensors for estimating symptom scores would classify ShareTrace as using multi-dimensional data.

Data Pre-processing *Raw data output* implies that no modification is made to the sensed data. *Filtering and denoising* entail "removing irrelevant and redundant data. In addition, they help to aggregate and make sense of data while reducing at the same time the volume to be stored" [9]. ShareTrace only retains the actor URIs that correspond to valid contacts (i.e., lasting at least 15 minutes).

Data Analytics *Machine learning (ML) and data mining analytics* are not real-time. They "aim to infer information, identify patterns, or predict future trends." On the contrary, *real-time analytics* consist of "examining collected data as soon as it is produced by the contributors" [9]. ShareTrace aligns with the former category since it aims to infer the infection risk of users.

Data Post-processing *Statistical post-processing* "aims at inferring proportions given quantitative examples of the input data. *Prediction post-processing* aims to determine "future outcomes from a set of possibilities when given new input in the system" [9]. ShareTrace applies predictive post-processing via risk propagation.

2.3.3 Communication Layer

Infrastructured Technology *Cellular* connectivity "is typically required from sensing campaign[s] that perform real-time monitoring and require data reception as soon as it is sensed." *WLAN* "is used mainly when sensing organizers do not specify any preferred reporting technologies or when the application domain permits to send data" at "a certain amount of time after the sensing process" [9]. Infrastructured technology is also referred to as the *infrastructured transmission paradigm* [30]. ShareTrace does not require cellular infrastructure, because it is delay-tolerant and thus only requires WLAN.

Infrastructure-less Technology *Infrastructure-less technologies* "consists of device-to-device (D2D) communications that do not require any infrastructure...but rather allow devices in the vicinity to communicate directly." Technologies include *WiFi-Direct*, *LTE-Direct*, and *Bluetooth* [9]. Infrastructure-less technology is also called the *opportunistic transmission paradigm* [30]. ShareTrace uses Bluetooth because of its energy efficiency and short range.

Upload mode With *relay uploading*, "data is delivered as soon as collected." *Store-and-forward* "is typically used in delay-tolerant applications when campaigns do not need to receive data in real-time" [9]. Because ShareTrace is delay-tolerant, it uses store-and-forward uploading.

Methelodgy *Individualized sensing* is "when each user accomplishes the requested task individually and without interaction with other participants." *Collaborative sensing* is when "users communicate with each other, exchange data[,] and help themselves in accomplishing a task or delivering information to the central collector. Users are typically grouped and exchange data exploiting short-range communication technologies, such as WiFi-[D]irect or Bluetooth...Note that systems that create maps merging data from different users are considered individual because users do not interact between each other to contribute" [9]. The methodology of sensing is similar to the *sensing* *scale* which is typically dichotomized as *personal* [16, 27] (i.e., individualized) and *community* [16] or *group* [27] (i.e., collaborative). ShareTrace is inherently collaborative, relying on mobile devices to exchange actor URIs to estimate infection risk. Thus, collaborative sensing is used.

Timing *Timing* is based on whether devices "should sense in the same period or not." *Synchronous timing* "includes cases in which users start and accomplish at the same time the sensing task. For synchronization purposes, participants communicate with each other." *Asynchronous timing* occurs "when users perform sensing activity not in time synchronization with other users" [9]. ShareTrace requires synchronous timing, because contact sensing inherently requires synchronous communication between the involved devices.

Sensor Deployment *Dedicated deployment* involves the use of "non-embedded sensing elements," typically for a specific task. *Non-dedicated deployment* utilizes sensors that "do not require to be paired with other devices for data delivery but exploit the communication capabilities of mobile devices" [9]. ShareTrace relies on non-dedicated deployment since it relies on Bluetooth that is ubiquitous in modern-day mobile devices.

Sensor Activity *Always-on sensors* "are required to accomplish mobile devices['] basic functionalities, such as detection of rotation and acceleration… Activity recognition [i.e., context awareness]…is a very important feature that accelerometers enable." *On-demand sensors* "need to be switched on by users or exploiting an application running in the background. Typically, they serve more complex applications than always-on sensors and consume a higher amount fo energy" [9]. ShareTrace uses Bluetooth, which may be considered on-demand. While energy efficient, users do control when it is enabled. Ideally, ShareTrace would also use always-on sensors to enable Bluetooth with context awareness (i.e., that the user is carrying or nearby the device).

Acquisition *Homogeneous acquisition* "involves only one type of data and it does not change from one user to another one," while *heterogeneous acquisition* "involves different data types usually sampled from several sensors" [9]. ShareTrace is homogeneous, because all users sense the same data from one type of sensor.

Sampling Frequency *Continuous sensing* "indicates tasks that are accomplished regularly and independently [of] the context of the smartphone or the user['s] activities." *Event-based sensing* is "data collection [that] starts after a certain event has occurred. In this context, an event can be seen as an active action from a user or the central collector, but also a given context awareness" [9]. ShareTrace sensing is continuous but would ideally be event-based to conserve device energy.

Sensing Responsibility When the *mobile device* is responsible, "devices or users take sampling decisions locally and independently from the central authority…When devices take sampling decisions, it is often necessary to detect the context [of the] smartphones and wearable devices…The objective is to maximize the utility of data collection and minimize the cost of performing unnecessary operations." When the *central collector* is responsible, they make "decisions about sensing and communicate them to the mobile devices" [9]. Given the human-centric nature of the ShareTrace sensing task, mobile devices are responsible.

User involvement *Participatory involvement* "requires active actions from users, who are explicitly asked to perform specific tasks. They are responsible to consciously meet the application requests by deciding when, what, where, and how to perform sensing tasks." *Opportunistic involvement* means that "users do not have direct involvement, but only declare their interest in joining a campaign and providing their sensors as a service. Upon a simple handshake mechanism between the user and the MCS platform, a MCS thread is generated on the mobile device (e.g., in the form of a mobile app), and the decisions of what, where, when, and how to perform the sensing are delegated to the corresponding thread. After having accepted the sensing process, the user is totally unconscious with no tasks to perform and data collection is fully automated…The smartphone itself is context-aware and makes decisions to sense and store data, automatically determining when its context matches the requirements of an application. Therefore, coupling opportunistic MCS systems with context-awareness is a crucial requirement" [9]. Earlier works on MCS refer to user involvement as the *sensing paradigm* [16, 27, 30]. ShareTrace is opportunistic, ideally with context-awareness.

Application	Task	Scheduling	Proactive	•
			Reactive	
		Assignment	Centralized	
			Decentralized	•
		Execution	Single task	•
			Multi-tasking	
	User	Recruitment	Voluntary	•
			Incentivized	
		Selection	Platform-centric	
			User-centric	•
		Type	Consumer	•
			Contributor	•
Data	Management	Storage	Centralized	
			Distributed	•
		Format	Structured	•
			Unstructured	
		Dimension	Single dimension	•
			Multi-dimensional	
	Processing	Pre-processing	Raw data	
			Filtering and denoising	•
		Analytics	ML and data mining	•
			Real-time	
		Post-processing	Statistical	
			Prediction	•
Communication	Technologies	Infrastructured	Cellular	•
			WLAN	•
		Infrastructure-less	LTE-Direct	
			WiFi-Direct	
			Bluetooth	•
	Reporting	Upload mode	Relay	
			Store and forward	•
		Methodology	Individual	
			Collaborative	•
		Timing	Synchronous	•
			Asynchronous	
Sensing	Elements	Deployment	Dedicated	
			Non-dedicated	•
		Activity	Always-on	•
			On-demand	○
		Acquisition	Homogeneous	•
			Heterogeneous	
	Sampling	Frequency	Continuous	•
			Event-based	○
		Responsibility	Mobile device	•
			Central collector	
		User involvement	Participatory	
			Opportunistic	•

Table 2.1: Classification of ShareTrace as a mobile crowdsensing (MCS) application. This table follows the four-layered architecture of MCS applications proposed by Capponi et al. [9]. Always (•); with context-awareness (○).

Chapter 3

Conclusion

In May 2023, the World Health Organization (WHO) declared that COVID-19 is no longer a “global health emergency” [46]. However, as is evident throughout history, the risk posed by emerging pathogens persists [36, 43]. Thus, research on effective approaches, such as contact tracing, to preventing and mitigating future outbreaks remains critically important.

Bibliography

[1] Gul Agha. "Actors: A model of concurrent computation in distributed systems". PhD thesis. MIT, 1985. URL: `http://hdl.handle.net/1721.1/6952` (cit. on pp. 10, 11, 14).

[2] Erman Ayday, Youngjin Yoo, and Anisa Halimi. "ShareTrace: An Iterative Message Passing Algorithm for Efficient and Effective Disease Risk Assessment on an Interaction Graph". In: *Proc. 12th ACM Con. Bioinformatics, Comput. Biology, Health Inform.* BCB 2021. 2021 (cit. on pp. 5, 7, 8, 10, 22, 25, 26).

[3] Erman Ayday et al. *ShareTrace: A Smart Privacy-Preserving Contact Tracing Solution by Architectural Design During an Epidemic.* White paper. Case Western Reserve University, 2020. URL: `https://github.com/cwru-xlab/sharetrace-papers/blob/main/sharetace-whitepaper.pdf` (cit. on pp. 10, 22, 25, 26).

[4] Alain Barrat and Ciro Cattuto. "Temporal Networks of Face-to-Face Human Interactions". In: *Temporal Netw.* Ed. by Petter Holme and Jari Saramäki. Underst. Complex Syst. Springer, 2013. DOI: `10.1007/978-3-642-36461-7\_10` (cit. on p. 20).

[5] Juan Benet. *IPFS - Content Addressed, Versioned, P2P File System*. 2014. DOI: 10.48550/arXiv.1407.3561. arXiv: 1407.3561 [cs.NI] (cit. on p. 25).

[6] Christopher M. Bishop. "Pattern Recognition and Machine Learning". In: *Inf. Sci. Stat.* Ed. by M. I. Jordan, Robert Nowak, and Bernhard Schoelkopf. Springer, 2006 (cit. on p. 6).

[7] Allan M. Brandt. "The History of Contact Tracing and the Future of Public Health". In: *American Journal of Public Health* 112.8 (2022), pp. 1097–1099. DOI: 10.2105/AJPH.2022.306949 (cit. on p. 1).

[8] Mark Briers, Marcos Charalambides, and Chris Holmes. *Risk scoring calculation for the current NHSx contact tracing app*. 2020. DOI: 10.48550/arXiv.2005.11057. arXiv: 2005.11057 [cs.CY] (cit. on p. 3).

[9] Andrea Capponi et al. "A Survey on Mobile Crowdsensing Systems: Challenges, Solutions, and Opportunities". In: *IEEE Commun. Surv. Tut.* 21.3 (2019), pp. 2419–2465. DOI: 10.1109/COMST.2019.2914030 (cit. on pp. 22–24, 26–32).

[10] William Clinger. "Foundations of actor semantics". PhD thesis. MIT, 1981. URL: http://hdl.handle.net/1721.1/6935 (cit. on p. 10).

[11] Thomas H. Cormen et al. *Introduction to Algorithms*. Fourth. The MIT Press, 2022 (cit. on p. 9).

[12] Meggan E. Craft. "Infectious disease transmission and contact networks in wildlife and livestock". In: *Phil. Trans. R. Soc. B* 370 (1669 2015). DOI: 10.1098/rstb.2014.0107 (cit. on p. 19).

[13] Leon Danon et al. "Networks and the Epidemiology of Infectious Disease". In: *Interdiscip. Perspect. Infect. Dis.* 2011 (2011). DOI: 10.1155/2011/284909 (cit. on p. 19).

[14] Joeri De Koster, Tom Van Cutsem, and Wolfgang De Meuter. "43 years of actors: a taxonomy of actor models and their key properties". In: *Proceedings of the 6th International Workshop on Programming Based on Actors, Agents, and Decentralized Control*. AGERE 2016. 2016, pp. 31–40. DOI: 10.1145/3001886.3001890 (cit. on p. 10).

[15] Erich Gamma et al. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley, 1995 (cit. on p. 11).

[16] Raghu K. Ganti, Fan Ye, and Hui Lei. "Mobile crowdsensing: current state and future challenges". In: *IEEE Commun. Mag.* 49.11 (2011), pp. 32–39. DOI: 10.1109/MCOM.2011.6069707 (cit. on pp. 29, 31).

[17] Alexander E. Gorbalenya et al. "The species severe acute respiratory syndrome-related coronavirus: classifying 2019-nCoV and naming it SARS-CoV-2". In: *Nature Microbiology* 5.4 (2020), pp. 536–544. DOI: 10.1038/s41564-020-0695-z (cit. on p. 1).

[18] Bin Guo et al. "Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm". In: *ACM Comput. Surv.* 48.1 (2015), pp. 1–31. DOI: 10.1145/2794400 (cit. on p. 22).

[19] Lea Hamner et al. "High SARS-CoV-2 Attack Rate Following Exposure at a Choir Practice – Skagit County, Washington, March 2020". In:

MMWR Surveill. Summ. 69 (19 2020). DOI: 10.15585/mmwr.mm6919e6 (cit. on p. 7).

[20] Carl Hewitt. "Viewing control structures as patterns of passing messages". In: *Artificial Intelligence* 8.3 (1977), pp. 323–364. DOI: 10.1016/0004-3702(77)90033-9 (cit. on pp. 10, 11).

[21] Carl Hewitt and Henry Baker. *Laws for communicating parallel processes*. Working paper. MIT Artificial Intelligence Laboratory, 1977. URL: http://hdl.handle.net/1721.1/41962 (cit. on pp. 10, 11).

[22] Carl Hewitt, Peter Bishop, and Richard Steiger. "A universal modular ACTOR formalism for artificial intelligence". In: *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*. IJCAI'73. 1973, pp. 235–245. URL: https://dl.acm.org/doi/10.5555/1624775.1624804 (cit. on p. 10).

[23] Petter Holme. "Modern temporal network theory: a colloquium". In: *Eur. Phys. J. B* 88 (9 2015). DOI: 10.1140/epjb/e2015-60657-4 (cit. on p. 19).

[24] Petter Holme and Jari Saramäki. "Temporal networks". In: *Phys. Rep.* 519 (3 2012). DOI: 10.1016/j.physrep.2012.03.001 (cit. on p. 19).

[25] Andreas Koher et al. "Contact-based model for epidemic spreading on temporal networks". In: *Phys. Rev. X* 9 (3 2019). DOI: 10.1103/PhysRevX.9.031017 (cit. on p. 19).

[26] Frank R. Kschischang, Brendan J. Frey, and Hans A. Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Trans. Inf. Theory* 47 (2 2001). DOI: 10.1109/18.910572 (cit. on p. 4).

[27] Nicholas D. Lane et al. "A survey of mobile phone sensing". In: *IEEE Commun. Mag.* 48.9 (2010), pp. 140–150. DOI: 10.1109/MCOM.2010.5560598 (cit. on pp. 29, 31).

[28] Trystan Leng et al. "The effect of notification window length on the epidemiological impact of COVID-19 contact tracing mobile applications". In: *Communications Medicine* 2.1 (2022). DOI: 10.1038/s43856-022-00143-2 (cit. on p. 6).

[29] Andrey Y. Lokhov et al. "Inferring the origin of an epidemic with a dynamic message-passing algorithm". In: *Phys. Rev. E* 90 (1 2014). DOI: 10.1103/PhysRevE.90.012801 (cit. on p. 19).

[30] Huadong Ma, Dong Zhao, and Peiyan Yuan. "Opportunities in mobile crowd sensing". In: *IEEE Commun. Mag.* 52.8 (2014), pp. 29–35. DOI: 10.1109/MCOM.2014.6871666 (cit. on pp. 28, 31).

[31] Robert McCune, Tim Weninger, and Greg Madey. "Thinking Like a Vertex: A Survey of Vertex-Centric Frameworks for Large-Scale Distributed Graph Processing". In: *ACM Comput. Surveys* 48 (2 2015). DOI: 10.1145/2818185 (cit. on p. 17).

[32] Cristina Menni et al. "Real-time tracking of self-reported symptoms to predict potential COVID-19". In: *Nat. Med.* 26 (7 2020). DOI: 10.1038/s41591-020-0916-2 (cit. on p. 3).

[33] James Moody. "The Importance of Relationship Timing for Diffusion". In: *Soc. Forces.* 81 (1 2002) (cit. on p. 19).

[34] Arvind Narayanan et al. *A Critical Look at Decentralized Personal Data Architectures.* 2012. DOI: 10.48550/arXiv.1202.4503. arXiv: 1202.4503 [cs.CY] (cit. on p. 25).

[35] Romualdo Pastor-Satorras et al. "Epidemic processes in complex networks". In: *Rev. of Mod. Phys.* 87 (3 2015). DOI: 10.1103/RevModPhys.87.925 (cit. on p. 19).

[36] Jocelyne Piret and Guy Boivin. "Pandemics Throughout History". In: *Frontiers in Microbiology* 11 (2021). DOI: 10.3389/fmicb.2020.631736 (cit. on p. 33).

[37] Francisco Pozo-Martin et al. "Comparative effectiveness of contact tracing interventions in the context of the COVID-19 pandemic: a systematic review". In: *European Journal of Epidemiology* 38.3 (2023), pp. 243–266. DOI: 10.1007/s10654-023-00963-z (cit. on pp. 1, 2).

[38] Joren Raymenants et al. "Empirical evidence on the efficiency of backward contact tracing in COVID-19". In: *Nature Communications* 13.1 (2022). DOI: 10.1038/s41467-022-32531-6 (cit. on p. 6).

[39] Leonie Reichert, Samuel Brack, and Björn Scheuermann. "A Survey of Automatic Contact Tracing Approaches Using Bluetooth Low Energy". In: *ACM Transactions on Computing for Healthcare* 2.2 (2021). DOI: 10.1145/3444847 (cit. on p. 1).

[40] Christopher S. Riolo, James S. Koopman, and Stephen E. Chick. "Methods and measures for the description of epidemiologic contact networks". In: *J. Urban Health* 78 (3 2001). DOI: 10.1093/jurban/78.3.446 (cit. on p. 19).

[41] Viktoriia Shubina et al. "Survey of Decentralized Solutions with Mobile Devices for User Location Tracking, Proximity Detection, and Contact Tracing in the COVID-19 Era". In: *Data* 5.4 (2020). DOI: 10.3390/data5040087 (cit. on p. 1).

[42] Sudhvir Singh et al. "How an outbreak became a pandemic: a chronological analysis of crucial junctures and international obligations in the early months of the COVID-19 pandemic". In: *Lancet* 398.10316 (2021), pp. 2109–2124 (cit. on p. 1).

[43] Syed Amin Tabish and Syed Nabil. "An Age of Emerging and Reemerging Pandemic". In: *Health* 14 (2022), pp. 1021–1037. DOI: 10.4236/health.2022.1410073 (cit. on p. 33).

[44] Dennis Trautwein et al. "Design and evaluation of IPFS: A storage layer for the decentralized web". In: *Proceedings of the ACM SIGCOMM 2022 Conference*. SIGCOMM '22. Association for Computing Machinery, 2022, pp. 739–752. DOI: 10.1145/3544216.3544232 (cit. on p. 25).

[45] Carmela Troncoso et al. "Systematizing Decentralization and Privacy: Lessons from 15 Years of Research and Deployments". In: *Proceedings on Privacy Enhancing Technologies* 2017.4 (2017), pp. 307–329. DOI: 10.1515/popets-2017-0056 (cit. on p. 25).

[46] Jacqui Wise. “Covid-19: WHO declares end of global health emergency”. In: *BMJ* 381 (2023). DOI: 10.1136/bmj.p1041 (cit. on p. 33).

[47] Tao Zhou et al. “Bipartite network projection and personal recommendation”. In: *Phys. Rev. E* 76 (4 2007) (cit. on p. 10).

[48] Na Zhu et al. “A Novel Coronavirus from Patients with Pneumonia in China, 2019”. In: *New England Journal of Medicine* 382.8 (2020), pp. 727–733. DOI: 10.1056/NEJMoa2001017 (cit. on p. 1).

[49] Lorenzo Zino and Ming Cao. “Analysis, Prediction, and Control of Epidemics: A Survey from Scalar to Dynamic Network Models”. In: *IEEE Circuits Syst. Mag.* 21 (4 2021). DOI: 10.1109/mcas.2021.3118100 (cit. on p. 19).