Sharing mHealth Data via Named Data Networking

Haitao Zhang
UCLA
zhtaoxiang@gmail.com

Zhehao Wang
UCLA
zhehao@remap.ucla.edu

Christopher Scherb
University of Basel
christopher.scherb@unibas.ch

Claudio Marxer
University of Basel
claudio.marxer@unibas.ch

Jeff Burke
UCLA
jburke@remap.ucla.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

ABSTRACT

This paper introduces NDNFit, a distributed mobile health (mHealth) application built to use the newly proposed Named Data Networking (NDN) architecture instead of TCP/IP. The design is inspired by the Open mHealth ecosystem. Open mHealth uses a traditional cloud-enabled mobile architecture, but aspires to provide users with direct control of how their personal health data is used by applications and shared with other users within the ecosystem. NDNFit names and secures users' health data directly using NDN network primitives, a more effective building block towards the ideal of user control than IP-based solutions. Its design illustrates that NDN's data-centric approach to networking can be a better fit than current networking approaches for mobile health applications, especially those that foreground individuals' control over their own data and, at the same time, target interoperability. This paper discusses the design and initial implementation of an NDNFit prototype, which offers end-users a mobile fitness tracking application. The paper identifies important differences between NDN and TCP/IP for mHealth, concluding with a discussion of future work and research opportunities.

CCS Concepts

•Networks → Network properties; •Computer systems organization → Distributed architectures;

Keywords

mHealth, Security, Named Data Networking

1. INTRODUCTION

Advances in personal mobile technology are enabling users to collect and share health-related information outside of clinical settings in unprecedented ways. Recent surveys suggest there are more than 13,000 health-related apps available to Apple iPhone users and more than 6,000 to Android users [3]. Millions now use smartphones to count steps, check heart rates and blood pressures, and share fitness activities. To achieve this functionality, most mHealth applications rely on cloud services built over TCP/IP networks to store and manage data. In doing so, they inherit an administratively centralized–if technically distributed–approach to sharing and managing access to health information that is collected from and owned by individuals.

ICN'16, September 26-28, 2016, Kyoto, Japan
© 2016 ACM. ISBN 978-1-4503-4467-8/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2984356.2984379

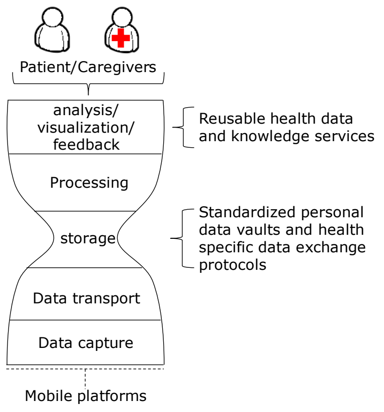
Patient/Caregivers
analysis/ visualization/ feedback
Processing
storage
Data transport
Data capture
Mobile platforms
Reusable health data and knowledge services
Standardized personal data vaults and health specific data exchange protocols

Figure 1: The data-centric hourglass Open mHealth architecture, adapted from [4].

Open mHealth [4] leverages the public's everyday mobile devices (e.g., smartphones and tablets) to extend evidence-based interventions beyond the reach of traditional care, and thereby improve disease management and prevention. Central to its Internet-inspired model is using *data exchange* as *the thin waist*, or common layer, of interoperability, as shown in Figure 1. Another crucial part of this vision is data exchange being user-controlled and privacy-aware across users, devices, applications, and vendor boundaries. However, this vision is challenging to achieve over existing cloud services that are built up from TCP/IP's host-centric approach to communication, which emphasizes hosts, sessions, and connections as the fundamental building blocks of distributed services.

NDN is a proposed future Internet architecture that makes exchange of named data packets the "thin waist" of information dissemination, in contrast to TCP/IP's host-to-host semantics. Further, NDN incorporates security building blocks at the network layer through signatures on every data packet. While a full description of the NDN architecture is outside the scope of this paper[1], even this basic description con-

[1]The reader is referred to [1] for more information.

tains a motivation for exploring Open mHealth on NDN. Both use named data for interoperability–Open mHealth as an application ecosystem, and NDN at the level of network architecture. Both seek to empower data owners to exercise control over their data without reliance on a particular secure channel, type of storage, or communications medium.

NDNFit is a prototype mHealth ecosystem designed and implemented to explore these alignments and investigate how NDN may be more effective than TCP/IP for certain types of modern applications. This paper provides preliminary results of this effort–an initial design, operational prototype (NDNFit), and discussions of what has been learned so far in creating an Open mHealth ecosystem on NDN.

2. DESIGN OF NDNFIT

NDNFit was designed to offer end-users a familiar fitness tracking application, and simultaneously serve as a prototype *mobile health ecosystem* that uses NDN to communicate. To limit the problem scope, it focuses specifically on physical activity data, e.g., records of walking or running collected in a consumer (non-clinical) context. From an end-user's perspective, NDNFit is currently experienced as a mobile and web-based application that 1) the user runs on their phone while exercising to capture and report time-location traces using GPS data; 2) manually or automatically classifies and segments the data to identify walking, jogging, and running activity; 3) reports activity statistics back to the user on a mobile website; and 4) provides identity management, data verification, and data access control features.

Unlike a traditional fitness application, NDNFit is designed to enable end-users to select from many possible components for data storage, processing, and visualization, each potentially operated by a different provider. Following the philosophy of Open mHealth, NDNFit aims to be an ecosystem of value-added services, from which users essentially *compose their own health data processing networks.* While one can aspire to this using the TCP/IP Internet, there are significant challenges. For example, scalable data storage and reliable access control are hard to achieve outside of cloud services from major providers, which are internally quite complex. Further, interoperability requires not only defining common data taxonomies and formats, but also mapping them to service and authentication APIs and specific IP endpoints, potentially across different providers. These dependencies and mappings, which become increasingly complex in large-scale deployments with real-world security requirements, limit innovation.

With NDNFit, the aim is to show that NDN's primitives simplify how a distributed application ecosystem can be brought to life, with lower barriers to entry for innovative services. NDNFit uses consistent data naming alone to achieve interoperability, rather than also having to define service APIs like Open mHealth's RESTful interfaces. It achieves authentication and access control of the data directly at the packet level, instead of inferring data authenticity from TLS connections. This reduces reliance on perimeter-based security and simplifies service chaining. Below, NDNFit's naming conventions are described, followed by brief descriptions of how named data is leveraged to support verification, access control, and distributed processing.

2.1 Namespace design

Figure 2 illustrates the NDNFit namespace, which was designed iteratively to meet several requirements: 1) to *name the data* from an application perspective, i.e., to name the physical activity samples themselves as the "thin waist" of interoperability; 2) to make common data requests efficient using only Interest-Data exchange; 3) to reflect the trust relationships between different system elements in data names; and 4) to enable name-based access control.

Our initial approach is to name the data in a way that makes sense for the application. The first two components, `/org/openmhealth`, identify the ecosystem and provide a trust anchor, `openmhealth/key`, for its components and, as will be seen in Section 2.4, also for users' health data. Each user has one or more unique identifiers in this namespace, e.g., `<user-id>`.[2] Each user namespace has four children: `key`, `devices`, `data`, and `read`, containing data on cryptographic identity, devices acting as health data sources, the (fitness) data samples themselves, and read-authorization data for name-based access control, respectively.

Fitness data consists of time-location series samples `data/fitness/physical_activity/time_location` named with their `timestamp`, which enable Interests to be easily constructed to retrieve data for any given time intervals. In cases when the number of samples that an application wishes to store for a given timestamp range exceeds the NDN packet size, the data is further split into named segments.[3] Following approaches in past participatory sensing work [2], these time-location traces are asynchronously annotated with additional metadata by various services in the ecosystem, an example of which is the `bout`, which describes a time range of activity samples that has been classified by a data processing unit as running, walking, etc.

Sections 2.3, 2.4, and 2.5 describe modifications and additions to the namespace to support efficient data transport for intermittently connected sources, identity and trust management, and data-centric access control, respectively.[4]

2.2 Application architecture

The NDNFit namespace names the data operated on by various components of the NDNFit prototype. Following the Open mHealth application architecture, NDNFit defines four main components, shown in Figure 3: **Capture Apps** that collect health data and publish it according to namespace and payload conventions described in the previous sections; **Data Storage Units (DSUs)** that are responsible for the persistent storage of users' health data; **Data Processing Units (DPUs)** that provide composable, value-added post-processing of that data; and **Data Visualization Units (DVUs)**, which enable the user to visualize and interact with their personal health data.

Users interact with the data via their mobile devices, which run the capture application (fitness data producer) as well as the "identity manager" and "authorization manager" to select signing keys and configure access control. These mobile devices also access DVUs that visualize and display fitness data from the NDNFit namespace.

One or more data storage units (DSUs) serve as repositories for users' health data. Conceptually, DSUs are simi-

[2]Services are similarly identified.

[3]Data payloads are outside of the scope of this paper, but typically JSON in our prototype.

[4]Supporting producer mobility and multiple DSU providers is outside this paper's scope, but will be addressed in future work, likely through techniques like those described in [12].

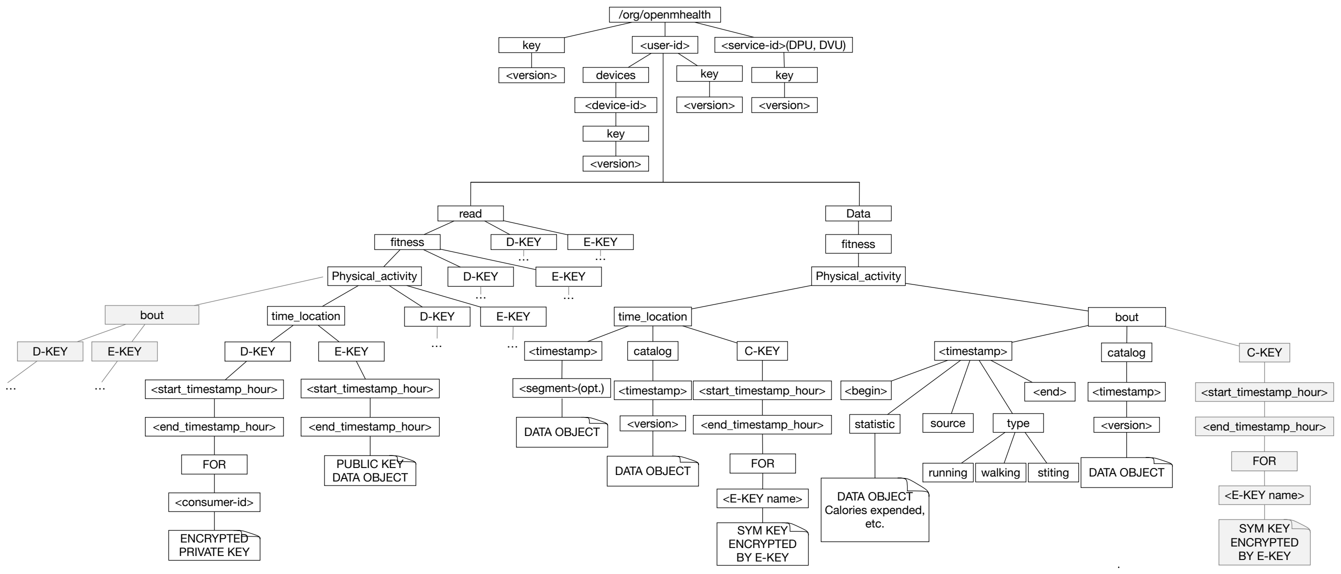

Figure 2: NDNFit namespace and data objects.

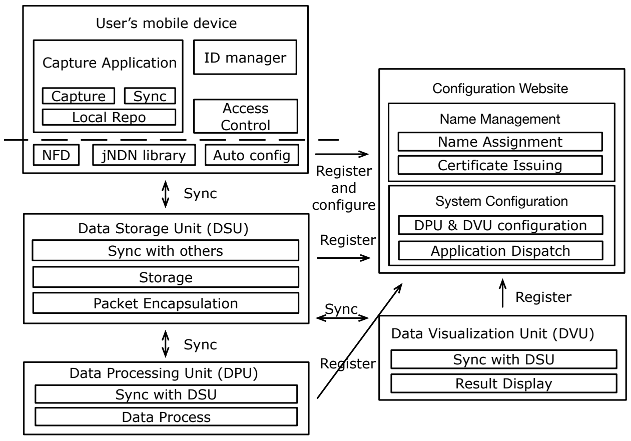

Figure 3: NDNFit application architecture.

lar to Personal Data Vaults (PDVs) described in [5]. DSU providers offer health data storage under a fiduciary responsibility to protect and not to profit from the user's data, and enable *selective sharing* with other users and services. Simplifying provisioning, interoperability, and selection of DSUs empowers users through more options–e.g., one could run a personal repo transparently replicated by a commercial data vault provider in a friendly jurisdiction.

Data processing units (DPUs) are entrusted by users to consume raw data and produce derived data on demand, such as the classification annotations in the `bout` namespace. Using Named Function Networking (see Section 2.6), DPUs produce derived data in response to requests from DVUs.

Finally, a set of processes manage the `/org/openmhealth` namespace, acting as a root of trust and assigning identities to users, DSUs, DPUs, and DVUs.[5]

[5]The administrative structure of these processes is an important open question; for now we envision a non-profit consortium that manages the namespace simply to avoid collision.

## 2.3 Data transport protocol

NDN employs a request-response model. Instead of relying on application-layer protocols to determine which host has desired data, consumers issue *Interests* to the network layer directly for named data and rely on the network to provide matching *Data* packets if available. NDNFit data consumers issue Interests in the namespace described in Section 2.1. In it, physical activity samples are named uniquely with the time they are generated. These timestamp-containing names are descriptive from an application perspective, allow asynchronous publishing (unlike sequence numbers), and enable random access simply by constructing appropriate Interest names. However, consumers may not know the exact timestamp they wish to retrieve, and it can be difficult to determine through Interest-Data exchange alone when *new* data become available within a time interval, e.g., as post-processed data become available asynchronously.

Thus, the namespace also includes a *data catalog* providing one level of indirection to consumers that can benefit from it. Catalogs are data packets similar to manifests [6], with the structure given in Figure 4 and naming shown in Figure 2. Like packets containing fitness samples, they are named with a timestamp, but are published by data producers at a predefined interval. For example, if the predefined interval is ten minutes, the catalog packet with ISO 8601 timestamp 20160101T100000 will contain the names of data packets produced between 20160101T100000 and 20160101T101000. Catalogs are versioned so they can be updated when new data for a given time interval are made available. Consumers wishing to use catalogs send Interests to fetch the catalog first, extract the data names, then issue further Interests for the constituent Data packets.

## 2.4 Identity and trust management

Authenticity of health data is critical. In most deployed services for consumer mobile health, including Open mHealth, data authenticity is established through interactive authentication to trusted servers, e.g., via TLS with implicitly

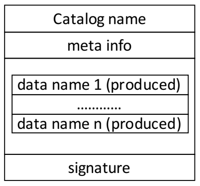

Figure 4: The catalog packet format.

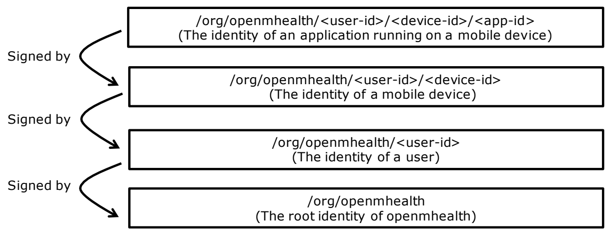
/org/openmhealth/<user-id>/<device-id>/<app-id>
(The identity of an application running on a mobile device)
Signed by
/org/openmhealth/<user-id>/<device-id>
(The identity of a mobile device)
Signed by
/org/openmhealth/<user-id>
(The identity of a user)
Signed by
/org/openmhealth
(The root identity of openmhealth)

Figure 5: Trust relationship for NDNFit application.

trusted certificates. This ties the trust in the data to the connection over which it is received, as opposed to making it inherent in the data itself [8]. This poses challenges for creating an interoperable ecosystem in which *sharing* authentic data is a key concern. These challenges can be addressed by data-centric security in NDN.

In NDNFit, the root of trust authorizes a user (or a service) to publish data under the `/org/openmhealth` namespace by signing a key the user controls. Keys are held in NDN named data objects using a standard certificate format that binds keys and namespaces. A user signs a device's key, and the device signs application keys with its own key to authorize the apps. Health data is acquired and published by authorized applications on authorized devices.

That is, processes producing health data, like the mobile capture app, sign the data they produce using keys from a trust chain with the user's key at the root. Applications, such as the DPU and DVU, can then verify the data's *validity* by traversing this chain–either to the user's root key, or all the way to the Open mHealth root key, depending on their trust policies–and caching the result for further use. The starting point to this chain is obtained from a data packet's *Key Locator* field. The hierarchical relationship used by NDNFit is shown in Figure 5. How NDN enables such relationships to be easily described and verified is discussed below.

Verifying data using this trust model leverages the NDN team's prior work in *schematized trust*[10], in which the relationship between data names and their expected signing key names can be expressed using regular expressions. NDNFit has a preconfigured trust schema that can be published as named, signed data in the application ecosystem, and then suggested or selected by services (or end-users), enabling data consumers to consult the schema and verify any received data consistently, regardless of where it comes from or where it is stored, eliminating the dependency on "connection-based" security. This model can be easily extended to the non-hierarchical, web-of-trust style models expected for users, which are not discussed here.

Section 3 describes how the NDNFit prototype manages the creation and assignment of certificates and keys.

2.5 Access control

In conversations with Open mHealth developers, OAuth-style authentication has been described as a significant pain point in its implementation over the existing Internet, especially when considering service chaining. NDN provides important, if still under development, advantages over such approaches. Per-packet signatures and schematized trust, described above, enables data verification without connection or session authentication. Similarly, encrypting data at creation enables access control to be achieved independently of how data is exchanged. *Name-based access control* (NAC) [11], which was developed with NDNFit as a potential use case, is adopted to test its applicability and effectiveness.

NAC provides a data-centric access control mechanism with the basic relationships between keys and data shown in Figure 6. Data is divided into minimum access units (MAUs) based on the acceptable granularity of access control. Each MAU is encrypted with a unique symmetric content key (`C-KEY`) when it is produced, for example, by the capture application or DPU. The data owner, which may be different from the data producer, generates a list of asymmetric key pairs (key-encrypt key `KEK` and key-decrypt key `KDK`), each of which is a *consumption credential* for accessing a given set of MAUs. The owner encrypts the `KDK` for every authorized consumer (e.g., a DPU or DVU) using that consumer's public key. Data producers fetch and use data owners' `KEK`s to encrypt the `C-KEY`s of the data covered by the consumption credentials' access authority. To read encrypted data, consumers fetch the data, `C-KEY`, and `KDK`, decrypting them in reverse order. More details, including the management of consumption credentials and duration of access, are described in [11].

NAC provides a building block to enable data owners to control access right as their data is produced, independently of how they are exchanged. Granting access to MAUs of user data requires no online service negotiation. Access is granted by encrypting data's `C-KEY`s for the consumption credentials whose access authority covers the data, and encrypting `KDK`s of those consumption credentials for authorized consumers.[6] This process is handled close to the network layer, using naming convention to deal with the problems of data signing, encryption, and key publishing, rather than in higher level protocols and services.[7] Users direct how data is encrypted by describing 1) relationships between data and consumption credentials and 2) for whom the consumption credentials' `KDK`s are encrypted. End-users can grant entities in the ecosystem–e.g., a DSU, DPU or DVU–access to their data directly by key naming, signing, and encrypting alone. At the same time, end-users can also delegate access management authority of part or all of their data to other services, which grant access to other entities according to user-defined rules–if users do not want to do it themselves.

The NDNFit prototype design incorporates NAC into each component, and implements it throughout, with some limitations in the DPUs described below. The namespace shown in Figure 2 incorporates the corresponding keys for data (`C-KEY` branches) and consumption credential (`KEK`s as `E-KEY` branches, and `KDK`s as `D-KEY` branches) at different levels in

[6]For now, this is done at the producing application or in a trusted DSU.

[7]More so than with schematized trust, how to present these capabilities in user interfaces poses some human-computer interaction (HCI) challenges that we are still exploring.

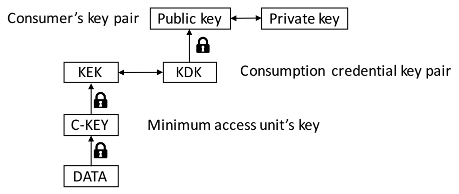

Figure 6: Key relationship in name-based access control.

the data hierarchy.

2.6 Named Functions and the DPU

Data capture in NDNFit is complemented by data derivation and aggregation in DPUs, which have the role of transforming captured data according to the user's requirements and depending on available processing functionality–e.g., offered by third parties. This requires an open framework where data is produced on demand and new processing functions are added over time by service providers. Named Function Networking (NFN) [7] achieves these requirements at the network level. In NFN, *processing expressions* become an interest's "name", for which the network has to produce the result in a matching Data packet. In other words, NFN's approach is *to name results*. Instead of using a single data reference, NFN operates on complex expressions which can reference named functions as well as (NDN or NFN) parameters. An NFN-enabled node is able to analyze such expressions, forward partial sub-expressions and combine retrieved intermediate results, pull executable code if the named function is mobile, or even directly compute the final result, depending on its capabilities. In the spirit of named data, results are expressed in a location-agnostic way that lets the named function network orchestrate the computation.

The name of a NFN-enhanced Interest consists of a routable prefix and a workflow definition. The routable prefix is the name of the input data or the name of the function call. Since NDN networks apply the longest prefix matching the Interest name, the Interest is routed to a node which has a copy of the Data or, in case of a function, is capable of executing it. The rest of the name, carrying the workflow, is not relevant for the forwarding decision.

The workflow itself is represented as a λ-expression. For example, given the input data `/fitness/data/run1` to compute the distance, one can "name the result" by writing `call /functionlib/distance /fitness/data/run1`. The syntactic construct of λ-abstraction (where the `@` replaces the λ) is used to prepend the input data name: `/fitness/data/run1 (@x call /functionlib/distance x)`. This results in a NDN name wherein the prefix points at the data (or the function) and having additional name components for the lambda abstraction and the workflow. Note that this mapping is bidirectional. The NFN-enhanced NDN name can be converted back into an expression that is equivalent to the original input.[8]

In NDNFit, the DPU also acts as a proxy between NDN and NFN as shown in Figure 7. It hides key handling and encryption/decryption of raw data according to NAC, and is responsible for tunneling NFN results that the requesting DSU will cache. For example, if a DSU receives an Interest

[8]In practice, additional rewriting prepends the ecosystem prefix to the function name and appends the suffix components corresponding to the arguments.

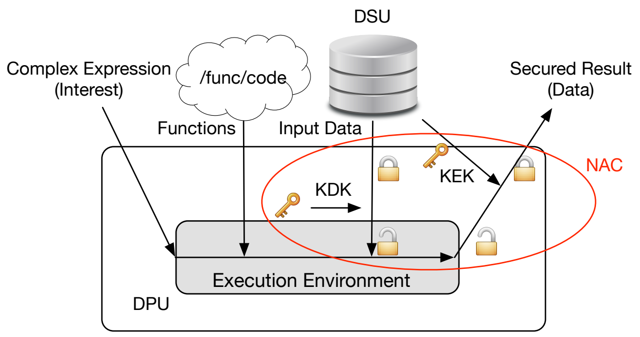

Figure 7: NDN and NFN interface, described in Section 2.6.

for some mHealth data that have not yet been produced and which has an ordinary NDN name, it *rewrites* this name into the corresponding NFN computation request and delegates execution to the DPU. The reply to this NFN query embeds the final result, which the DSU extracts and puts into its store. This mechanism enables the NFN back-end entity to sign the result and bind it *to the front-end NDN name*, instead of the NDN-encoded NFN name by which the result was requested. A future version of NDNFit will provide NFN-native support for NAC.

3. IMPLEMENTATION

The above design, including authentication via schematized trust and encryption-based access control, has been implemented in a proof-of-concept version of NDNFit. A mobile data capture application and *Identity Manager* application are implemented using Java, and run on the Android platform. A DSU is implemented in C++ and a DPU is implemented using Scala, both on the Linux platform. A DVU is implemented using JavaScript, and was tested within the Chrome web browser.

The NDNFit implementation also includes an identity system, in which a certificate authority is the trust anchor and is responsible for authorizing users and components through a web interface. The authority website is implemented using Python, and runs on the Linux platform. End-users run the *Identity Manager* to authorize applications and configure trusted DSUs, DPUs and DVUs using the mechanisms described in Section 2. Certificates are distributed as named data, and they can be served by corresponding nodes, or by the DSUs. The data capture application, upon first launch, requests a chosen user identity to sign its certificate. The identity manager receives these requests, and requests the user makes a decision. Once the initial setup is finished, the user can use the data capture application to produce data. When the mobile device has a network connection, the DSU will fetch data and store it.[9] Lastly, the DPU and DVU fetch data and process or display it as needed.

4. DISCUSSION

The existing Open mHealth reference application, Ohmage[9], uses a traditional cloud-based approach featuring web services endpoints, OAuth authentication, and relational database backends. Data exchange is standardized through a combi-

[9]For redundancy, we implement DSU replicas, synchronized using ChronoSync[13].

nation of web APIs, data names, and data formats. With Ohmage's approach and the NDNFit team's experience in mind, the benefits of NDN and the open challenges for mobile health are briefly described below.

Names have significant power in NDN. NDNFit uses data names to organize data access, identity and trust management, distributed processing, and access control. Using NDN allows these application-defined data names to be operated on at the network layer, unifying application, protocol, and network behavior in a way that can't be achieved in TCP/IP. Our design practice in NDNFit suggests that namespace design is of primary importance, and can be used to drive development of each aspect of secure communications.

Named data simplifies protocols and security in a data-centric ecosystem. To provide health data in the NDNFit ecosystem, an application must obtain the appropriate keys, which can be done through standard Interest-Data exchange, naming, signing, and encrypting the data appropriately, and making it available on the network. To consume health data, an application must obtain the appropriate trust schema and decryption keys, and implement the necessary pattern of Interests. While, for now, each step in this process is new territory, there are already significant simplifications over implementing similar functionality using a standard TCP/IP web services approach, especially when data provenance and granular access control are required. Further, in NDN, various communication functions are unified through expressive naming schemes visible to the network layer. As such, health data, trust models, and access control schemes can all be easily fetched using the same primitives supported by any NDN network.

Robustness to diverse communication situations is more inherent. NDN provides intrinsic data caching, multicast, multi-path forwarding, and disruption recovery mechanisms. mHealth applications built on NDN should not need to address these at the application layer, which gives them the potential of being more efficient and robust than those built on IP.

Open challenges. Further challenges remain in this work. Many map to larger NDN research challenges, such as choosing the right naming conventions, in particular how best to balance the conflicts between application and network's preferences on naming giving it is now shared between the two. Some involve more formally evaluating the benefits offered by building application over NDN. Others involve how to make new approaches easily grasped and debugged by developers, and communicated to users. NAC is an example of this; it provides significant power to NDNFit, but requires work in making end-user configuration possible and application development straightforward.

5. CONCLUSION

Motivated by Open mHealth's vision of an interoperable ecosystem of health data processing in which users have ultimate control over their data, NDNFit is a demonstration application that illustrates potential benefits of the NDN architecture for mobile health. Specifically, its design suggests the power of networking via application-named data, the usefulness of naming conventions to simplify application architecture. The design also demonstrates the power of securing data directly, which removes dependencies and constraints emerging from relying on underlying transport layers for security.

Acknowledgements

Many people contributed to NDNFit design discussions; the authors particularly thank Yingdi Yu, Van Jacobson, Alex Afanasyev, and Alex Halderman for their security-related contributions. This work is supported by the National Science Foundation through award CNS-1345318 and others.

6. ADDITIONAL AUTHORS

Additional authors: Christian Tschudin (University of Basel, email: `christian.tschudin@unibas.ch`).

7. REFERENCES

[1] A. Afanasyev, J. Burke, L. Zhang, K. Claffy, L. Wang, V. Jacobson, P. Crowley, C. Papadopoulos, and B. Zhang. Named Data Networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[2] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *Proc. Workshop on World Sensor Web at SenSys (WSW'06), Oct.*, 2006.

[3] D. S. Eng and J. M. Lee. The promise and peril of mobile health applications for diabetes and endocrinology. *Pediatr Diabetes*, 14(4):231–238, 2013.

[4] D. Estrin and I. Sim. Open mHealth architecture: an engine for health care innovation. *Science*, 330(6005):759–760, 2010.

[5] J. Kang, K. Shilton, D. Estrin, and J. Burke. Self-surveillance privacy. *Iowa L. Rev.*, 97:809, 2011.

[6] I. Moiseenko. Fetching Content in Named Data Networking with Embedded Manifests. Technical report, NDN Technical Report NDN-0025, 2014.

[7] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin. An Information Centric Network for Computing the Distribution of Computations. In *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, pages 137–146, 2014.

[8] D. Smetters and V. Jacobson. Securing network content. Technical report, 2009.

[9] H. Tangmunarunkit, C. Hsieh, B. Longstaff, S. Nolen, J. Jenkins, C. Ketcham, J. Selsky, F. Alquaddoomi, D. George, J. Kang, et al. Ohmage: A general and extensible end-to-end participatory sensing platform. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):38, 2015.

[10] Y. Yu, A. Afanasyev, D. Clark, V. Jacobson, L. Zhang, et al. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd Conference on Information-Centric Networking*. ACM, 2015.

[11] Y. Yu, A. Afanasyev, and L. Zhang. Name-Based Access Control. Technical report, NDN Technical Report NDN-0034, 2015.

[12] Y. Zhang, A. Afanasyev, J. Burke, and L. Zhang. A survey of mobility support in named data networking. In *Proceedings of the third Workshop on Name-Oriented Mobility (NOM 2016)*.

[13] Z. Zhu and A. Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *21st IEEE International Conference on Network Protocols*, pages 1–10, 2013.