Poster Abstract: Indoor Path Planning and Decentralized Access Control in Commercial Buildings

Leepakshi Bindra, Kalvin Eng, Yancheng Ou, Omid Ardakanian, Eleni Stroulia
{leepaksh,kalvin.eng,ou2,ardakanian,stroulia}@ualberta.ca
University of Alberta, Canada

ABSTRACT

We describe a methodology for indoor path planning and controlling access to different spaces, equipment sensing and control points in a commercial building. We implement three services for sensitivity-cost quantification of building spaces, path planning, and decentralized access control through smart contracts. These services rely on information that is captured in the Brick and BOT models of a building. We develop a calendar application on top of these services to show that the proposed system can greatly reduce the administration overhead in a real commercial building while providing fine-grained access control.

CCS CONCEPTS

• **Security and privacy → Security services**; • **Computer systems organization → Embedded and cyber-physical systems**;

ACM Reference Format:
Leepakshi Bindra, Kalvin Eng, Yancheng Ou, Omid Ardakanian, Eleni Stroulia. 2019. Poster Abstract: Indoor Path Planning and Decentralized Access Control in Commercial Buildings. In *The 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '19), November 13–14, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3360322.3361013

1 PROBLEM DEFINITION

Modern commercial buildings are complex distributed systems comprised of hundreds of sensors and actuators, which are located in different building spaces, and monitor or control various pieces of equipment. These sensing and control devices may impact building areas beyond the room in which they are physically located; for example, a thermostat located in a room can determine the temperature setpoint of multiple adjacent rooms, as described in [3]. Thus, unauthorized access to building spaces could have serious implications. This highlights the importance of controlling access to building spaces and equipment at the same time.

Managing permissions to access building spaces, read sensor measurements, write setpoint values, and delegate trust is an increasingly complex problem, especially in a commercial building that houses multiple organizations and has numerous long-term occupants and visitors. As a standard practice today, long-term building occupants are given access privileges to rooms and equipment based on their organizational roles, while visitors have to be escorted by their hosts. This approach is conservative and inflexible.

BuildSys '19, November 13–14, 2019, New York, NY, USA

ACM ISBN 978-1-4503-7005-9/19/11…$15.00
https://doi.org/10.1145/3360322.3361013

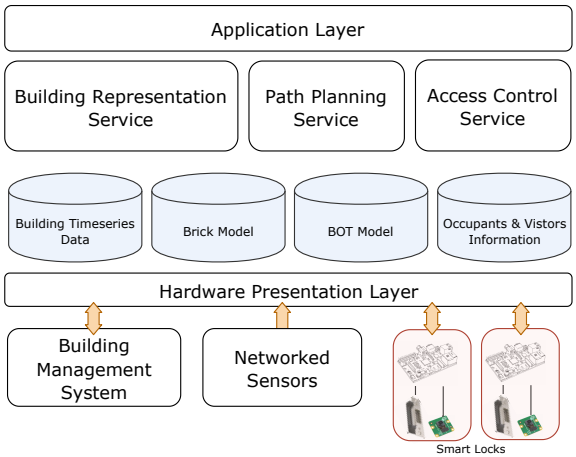

Figure 1: Overall System Architecture

In this poster abstract we describe the design and implementation of a set of services that enable smart building applications to list the paths between two building locations, quantify the cost-sensitivity of each path, and manage people's access privileges (to rooms and equipment points therein) in a fine-grained and decentralized fashion. We argue that these services make access control auditable and more flexible, and greatly reduce the administrative overhead by eliminating the reliance on a central authority. We illustrate the proposed method through an application scenario that involves inviting a group of people to a meeting that will take place in a specific room of a real office building.

Similar to [1], our access-control methodology relies on blockchain smart contracts to describe, grant, audit, and revoke fine-grained permissions for building occupants and visitors in a decentralized fashion. The smart contracts are specified through a process that leverages the information compiled from the BOT [5] and Brick [2] models of the building's spatial structure, equipment, and their relations. This information enables our methodology to grant an individual with just the right access privileges to let them reach their destination within the building.

2 IMPLEMENTATION

Figure 1 shows various operating system services described in [4] and different data stores containing information about a building's occupants, spaces, and equipment. The Hardware Presentation Layer resides on top of the Building Management System (BMS), and other sensors and actuators installed in the building, e.g., a thermostat or a smart lock that opens a door. This layer consists of many drivers that are necessary to interact with sensors and actuators. The BOT [5] and Brick [2] models describing the building's spatial structure and functional relationships between equipment

are stored in an RDF database. We implemented three services that rely on these models as described below:

- **Building-Representation Service** determines the cost of an indoor-path using information captured in the Brick model.
- **Path-Planning Service** relies on the aligned BOT and Brick model to find all possible paths between two given locations and the equipment housed in a location. These paths and equipment are presented to the user along with their costs, enabling them to choose a desired path based on its sensitivity-cost.
- **Access-Control Service** uses smart contracts to grant or revoke access, given a delegator, a delegate, a path corresponding to a sequence of building spaces and doors between them, and a time period during which the delegate should be able to access the resources on this path. It also validates the delegate's authority to access a resource at run-time.

2.1 Path-Planning & Building-Representation Services

The Path-Planning service uses the BOT and Brick models of the building are aligned together to create a unified RDF graph, which is queried via SPARQL. This RDF graph represents the relationships and entities that are present in both Brick and BOT. The Brick model captures information about equipment points (setpoints, sensors, etc.), their locations, and relationships between them. However, it does not describe adjacencies between different locations; this information is crucial to identify an indoor-path. The BOT model represents spatial information about a building using containment and adjacency relationships.

We ensure that the two models are oriented by aligning the entities corresponding to the same building location (e.g., room) in Brick and BOT model triples. This allows us to reason about which pathway enables access to which location. Syntactic entities that represent the same semantic entity in the two graphs are joined to create a new entity which is a subclass of Location in Brick and Space in BOT.

Pathways in the building are identified using a graph traversal algorithm (e.g., depth-first search) in the Path-Planning service with each path having a cost-sensitivity returned by the Building-Representation service. This cost-sensitivity is defined as:

$$\text{cost}(path) = \sum_{r \in path} \text{cost}(r) \tag{1}$$

$$\text{cost}(r) = \text{sensitivity}(r) + \sum_{p \text{ hasLocation } r} \text{weight}(p) \times (1 + \text{control}(p)) \tag{2}$$

where $path$ is a sequence of rooms r, sensitivity(r) is the numerical value of security zone classification for room r, p is a point (e.g., a setpoint or a sensor) that is part of a subsystem, weight(p) is the weight given to point p (determined using Analytic Hierarchy Process), and control(p) is the number of locations or zones affected by point p.

The Path-Planning service is implemented in Python using the RDFLib library to manage the graph and the Flask web framework to develop the API. In order to calculate the cost-sensitivity, we use SPARQL queries to obtain the following:

- the sequence of adjacent *rooms* leading from one location to another, e.g., the main building door to the meeting room;
- the set of *points* located in a room, their types, and the locations they influence through control relationship.

2.2 Access-Control Service

The Access-Control service runs on top of a private blockchain using the Ethereum network. Smart contracts deployed over this private blockchain are used to store the users as entities and access rules as relationships between the nodes to form a graph representing authorization. A Flask web API interacts with the nodes on the private blockchain to send transactions to the smart contracts.

There are three smart contracts which manage the access rules to define fine-grained permissions for a duration of time. Access is defined or revoked by following the rules described in these contracts. Apart from this, when a user tries to access any resource, the state of the contract is read to verify if the user has the required access to the requested resource. Updates to meeting schedules are handled by changing the state of the smart contract.

The *Archives* contract manages the entities and access rules for each user. Creation of entities or access rules is done using functions of this contract, transaction to which causes a state change. The state of the Archives contract is read when a request to verify access is received. The *Implications* contract cross-verifies the validity of access rules and processes the path resources that should be accessible to the delegate. Creating a new access rule leads to a state change, which is read during access verification. The *Exclusions* contract executes transactions when the delegator chooses to provide a list of resources that the delegate is not allowed to access, even though they are in spaces that are accessible to the delegate.

2.3 Calendar Application

We develop a calendar application that relies on the three services described above. This application parses an *iCalendar* (ics) file the meeting host uploads, extracts the meeting host and participants email addresses as well as the meeting time and duration and interacts with the access-control service to verify their identity. It then asks for paths to the meeting room from the path-planning service and prompts the meeting host to select a path from the set of possible paths based on their costs. Access permissions are created upon the selection of path and resources. A unique access QR code is generated for each participant which encodes the event unique ID extracted from the ics file, the host identity, and the participant identity, which is emailed to the meeting participant.

When a meeting participant arrives in the building and scans their QR code at a door on the selected pathway, the smart lock interacts with the calendar application, which further checks the time-restricted access permission with the access-control service. The access will be given by the smart lock once the identity is validated; it operates an electric door strike to open the door.

REFERENCES

[1] Michael P Andersen et al. 2018. Democratizing authority in the built environment. *ACM Transactions on Sensor Networks (TOSN)* 14, 3-4 (2018), 17.

[2] Bharathan Balaji et al. 2018. Brick: Metadata schema for portable smart building applications. *Applied Energy* 226 (2018), 1273–1292.

[3] Leepakshi Bindra et al. 2019. Decentralized access control for smart buildings using metadata and smart contracts. In *Proceedings of the 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems*. IEEE Press, 32–38.

[4] Stephen Dawson-Haggerty et al. 2013. BOSS: Building Operating System Services. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation*. USENIX, 443–457.

[5] Mads Holten Rasmussen et al. 2017. Recent changes in the building topology ontology. In *LDAC2017-5th Linked Data in Architecture and Construction Workshop*.