EPICGen: An Experimental Platform for Indoor Congestion Generation and Forecasting

Chrysovalantis Anastasiou
University of Southern California
Los Angeles, California, USA
canastas@usc.edu

Constantinos Costa
University of Pittsburgh
Pittsburgh, Pennsylvania, USA
costa.c@cs.pitt.edu

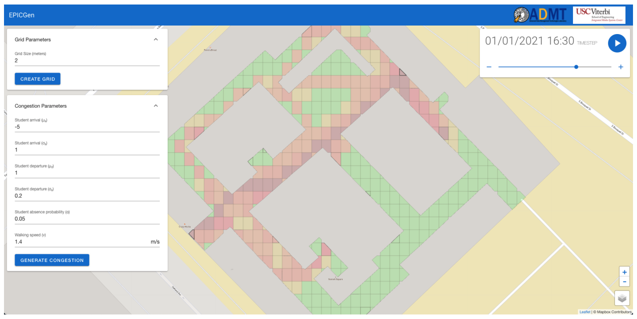
Panos K. Chrysanthis
University of Pittsburgh
Pittsburgh, Pennsylvania, USA
panos@cs.pitt.edu

Cyrus Shahabi
University of Southern California
Los Angeles, California, USA
shahabi@usc.edu

ABSTRACT

Effectively and accurately forecasting the congestion in indoor spaces has become particularly important during the pandemic in order to reduce the risk of exposure to airborne viruses. However, there is a lack of readily available indoor congestion data to train such models. Therefore, in this demo paper we propose *EPICGen*, an experimental platform for indoor congestion generation to support congestion forecasting in indoor spaces. *EPICGen* consists of two components: (i) *Grid Overlayer*, which models the floor plans of buildings; and (ii) *Congestion Generator*, a realistic indoor congestion generator. We demonstrate *EPICGen* through an intuitive map-based user interface that enables end-users to customize the parameters of the system and visualize generated datasets.

PVLDB Reference Format:
Chrysovalantis Anastasiou, Constantinos Costa, Panos K. Chrysanthis, and Cyrus Shahabi. EPICGen: An Experimental Platform for Indoor Congestion Generation and Forecasting. PVLDB, 14(12): 2803 - 2806, 2021.
doi:10.14778/3476311.3476349



Grid Parameters
Grid Size (meters): 2

Congestion Parameters
Student arrival (μ_a): -5
Student arrival (σ_a): 1
Student departure (μ_d): 1
Student departure (σ_a): 0.2
Student absence probability (α): 0.05
Walking speed (v): 1.4 m/s

Figure 1: The *EPICGen* data exploration user interface visualizes multiple timesteps of the congestion using an interactive time slider. It depicts the *Building Grid Layout* on a map and colors the cells based on the congestion density in the cell (i.e., green cells have low to no congestion, red cells are highly congested). Cells that contain doors (i.e., either entrances/exits or room doors) have a thicker outline.

1 INTRODUCTION

Modeling human mobility is a challenging task. One of the challenges is that while the underlying semantics of trajectories do not vary widely between different individuals [3], each trajectory remains very unique [7, 8]. Modeling human mobility is also an essential task for many applications, including congestion forecasting, which has become particularly important for indoor spaces during the current COVID-19 pandemic. According to the US Centers for Disease Control and Prevention (CDC), moving indoors through congested pathways dramatically increases the risk of exposure to airborne viruses[1] and, therefore, accurately forecasting the congestion in buildings is crucial for *Mobile Contact Avoidance Navigation (MCAN)* applications [6, 9].

MCAN applications rely on indoor congestion forecasting models to effectively recommend paths that minimize the exposure risk to airborne viruses. Congestion forecasting models, in turn, rely on repositories of historical congestion data to accurately model and forecast the congestion in buildings [1, 2]. However, such data repositories are not readily available. To this end, we develop an experimental platform, dubbed *EPICGen*[2], that implements an efficient algorithm for generating realistic indoor congestion datasets.

EPICGen consists of two components: (i) the *Grid Overlayer* algorithm, which models building floor plan elements, e.g., corridors, doors, rooms; and (ii) the *Congestion Generator*, which generates realistic indoor datasets given a building layout by simulating the indoor trajectories of individuals. Additionally, *EPICGen* provides an intuitive map-based user interface that allows end-users to customize the parameters required for the generator, and visualize the generated datasets as shown in Figure 1.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476349

[1]Centers for Disease Control and Prevention: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/how-covid-spreads.html

[2]EPICGen: An Experimental Platform for Indoor Congestion Generation.

The **contributions** of this demo paper are summarized as follows:

- We propose a novel partitioning scheme and algorithm, called *Grid Overlayer*, that enables granular modeling of congestion in buildings.
- We propose a method to generate realistic indoor congestion datasets using an efficient algorithm, dubbed *Congestion Generator*.
- We develop an intuitive map-based prototype system that allows the end-user to parameterize the algorithms and generate synthetic congestion datasets for any building on-the-fly.
- We showcase how our MCAN system, called *HealthDist*, uses *EPICGen* to train congestion forecasting models in order to effectively recommend paths through indoor buildings while minimizing the exposure to congested areas.

The remainder of the paper is structured as follows: Section 2 provides an overview of the system. Section 3 presents the demonstration artifact and scenarios.

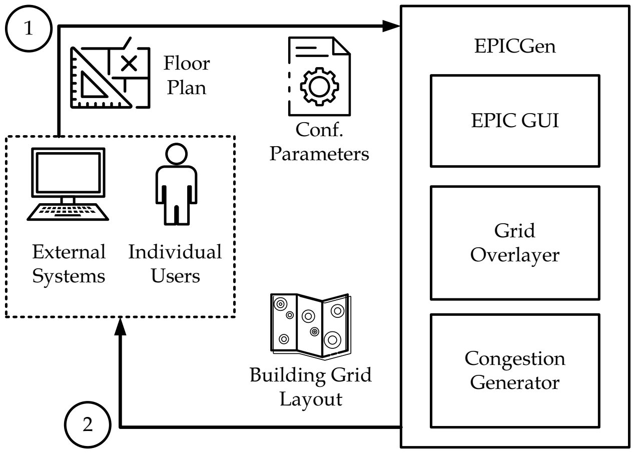


Figure 2: *Overview of the EPICGen system showcasing the order in which the components are invoked to generate realistic congestion datasets.*

2 SYSTEM OVERVIEW

EPICGen consists of two main components: (i) the *Grid Overlayer* described in Section 2.1; and (ii) the *Congestion Generator* described in Section 2.2. The *Grid Overlayer* is responsible for converting floor plans into a data structure, the *Building Grid Layout* that can model the congestion at a high spatial resolution. The *Generator* is responsible for generating realistic congestion data given a *Building Grid Layout* and simulation parameters.

Figure 2 illustrates an overview our *EPICGen* system. The first step is to invoke the *Grid Overlayer*. Subsequently, the output of the *Grid Overlayer* and a set of parameters are used to invoke the *Congestion Generator* component. *EPICGen* extracts floor plans from a variety of data sources and maintains a database in the back-end. End-users can interact with the system, select floor plans, and configure the algorithms through *EPIC GUI*, an intuitive map-based interface.

2.1 Building Grid Layout

In order to model the congestion inside buildings at a granular level, we develop the *Grid Overlayer* algorithm that divides the building corridors into smaller segments. The benefit of this approach is that each segment can be independently modeled, hence leading to higher resolution datasets and visualization.

Given a floor plan of a building in a semi-structured format (e.g., GeoJSON), the *Grid Overlayer* algorithm overlays a uniform grid over the building and detects those cells that overlap with corridors, doors, and rooms generating a *Building Grid Layout* structure. The *Building Grid Layout* does not retain every cell of the initial grid, i.e., cells that do not overlap with the floor plan of the building are discarded. Additionally, the cells are further split so that each cell overlaps exactly one floor plan element.

The *Grid Overlayer* algorithm works as follows. First, the floor plan of the building is spatially indexed, i.e., corridors (polygons), rooms (polygons), and door locations (points) are loaded into a spatial index, specifically an R-tree. A uniform grid that covers the entire building is generated with a configurable cell size ϕ. Next, the grid cells are loaded into a second R-tree. Finally, a join operation between the floor plan elements and the grid cells is performed using the previously constructed spatial indexes. Cells that overlap with at least one floor plan element, i.e., corridor, room, door, are qualified for post-processing while all other cells are discarded. During post-processing, the shape of the cells is adjusted to exactly fit their intersection area with the floor plan elements. If a cell overlaps more than one element, it is divided at the intersections of the elements it overlaps.

Figure 3 illustrates an example of *Grid Overlayer*'s output. On the left, the floor plan of the Sennot Square building at the University of Pittsburgh is depicted. The corresponding generated *Building Grid Layout* is shown on the right. Corridor cells are colored in gray and outlined with dark gray lines. Other types of cells are omitted for brevity.

2.2 Indoor Congestion Data Generator

Our real-world observations show that traffic inside academic buildings can be separated into two main categories: *scheduled* and *pass-through*. This observation is also confirmed by the study in [4]. Scheduled traffic is the traffic that follows a specific schedule, e.g., class schedule. This means that short time before and short time after a class, a spike of traffic appears in the building. Pass-through traffic is generated by people who enter the building in one door only to wander and exit at another. Therefore, we design and implement a congestion generator that addresses these two categories of traffic. To achieve this, congestion is generated in two phases. The first phase generates the pass-through traffic (*browsers*) whereas the second phase generates the scheduled traffic (*commuters*). Both phases require the *Building Grid Layout* (Section 2.1) as input along with a date range of the simulation.

Phase 1 (Pass-through traffic): The rate of congestion generated by this type of traffic varies depending on the time of day. For example, there is almost zero traffic during nighttime but a lot of people will pass through the building during the daytime. Hence,

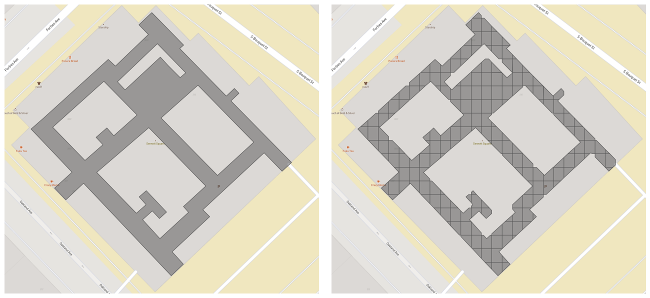


Figure 3: ***Floor plan of the Sennot Square building at the University of Pittsburgh campus (left) converted to a Building Grid Layout (right).***

the generator receives an additional input for this phase, namely the arrival rate of people at different times of the day. For every time step, the generator will randomly sample the number of people that arrive at each building door and for each person a destination exit door is selected. Then, assuming a constant walking speed (e.g., the average walking speed is 1.4 m/s), the trajectory of each person is simulated following the shortest path that is formed using corridor cells. The congestion of each cell is updated accordingly.

Phase 2 (Scheduled traffic): To simulate the scheduled traffic we require some sort of schedule for every room in the building, i.e., the time classes start and end for different days of the week as well as the audience size of each class. Hence, the generator accepts this schedule as an input for this phase. Besides the schedule, the generator requires an additional set of parameters, i.e., arrival rate of students before the class begins, departure rate after the class ends, and probability that a student is absent from class.

We simulate the arrival of students as a Gaussian distribution with mean μ_a and standard deviation σ_a that samples the number of minutes before the start time of the class the student enters the building. Similarly, we simulate the departure of students from class as a Gaussian distribution with mean μ_d and standard deviation σ_d that samples the number of minutes after the end time of the class the students leave the classroom. We also simulate the absence of a student from class as a binomial distribution with $p = \alpha$. Lastly, an average walking speed v is assumed as above.

Although we introduce the proposed generator in the context of academic buildings, the process can be generalized to almost any kind of building. The only hard requirement is the floor plan of the building. All other parameters ($\mu_a, \sigma_a, \mu_d, \sigma_d, \alpha, v$) can be provided by the user or estimated using other datasets, such as Wi-Fi access point connection logs. For example, access point log files, which are extracted from the network infrastructure of the University of Southern California (USC), are analyzed to estimate and predict the congestion in buildings during the pandemic in order to reduce the spread of the virus. The logs tend to show a spike in the number of connections a few minutes before the start of a class, and a drop in the number of connected users right after a class finishes. These spikes and drops can be correlated with the schedule in order to estimate the parameters $\mu_a, \sigma_a, \mu_d, \sigma_d$.

The scheduled traffic generator algorithm iterates over every schedule item and generates the respective traffic. For every student

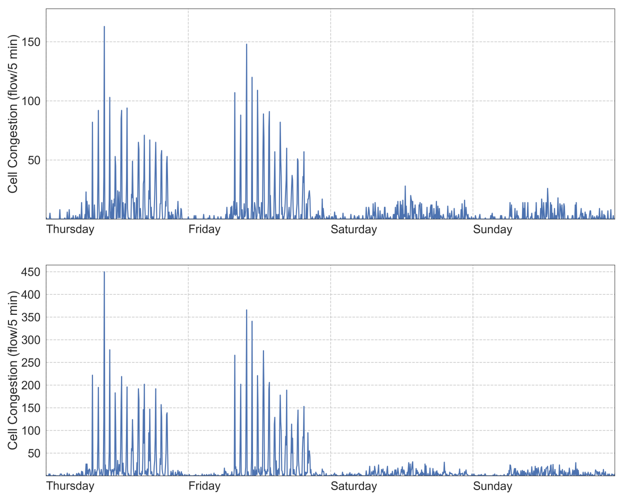


Figure 4: ***Generated congestion for a corridor cell aggregated into 5-minute intervals corresponding to Entrance 1 (top) and at a corridor intersection (bottom) of Wesley W. Posvar Hall at University of Pittsburgh.***

in the class, we first draw a sample from the absence distribution to decide if the student is attending the class. If the student is attending, we additionally draw a sample from the arrival distribution to decide how early that student is going to enter the building. An entrance door is selected uniformly at random and the student's trajectory to the respective room is generated using a constant speed of $v\ m/s$. The congestion of the cells that intersect with the student's path is updated accordingly. A similar process is employed for the departure of students from the class. For every student that was present, a departure time and an exit door are sampled and the student is routed while the congestion of cells that intersect with the path is updated. The final output is the congestion time series of every corridor cell aggregated in 5-minute intervals.

Schedule Generator: In the case that a schedule does not exist for a building or if the end-user of our system is interested in benchmarking a variety of scenarios, we propose a method to generate one. Three input parameters are required: (i) the audience size distribution, which is assumed to be a Gaussian distribution with mean μ_s and standard deviation σ_s; (ii) a set of event durations (class, no class) along with their respective probabilities, which are modeled as multinomial distribution; and (iii) the probability p_c that an event is a class or not. Optionally, the time of day when classes begin and end, e.g., from 7AM to 9PM, can be provided. Starting at the time that the schedule is configured to begin, e.g., 7AM, we sample two pieces of information, namely the duration of the next event δ and whether the next event is a class or not. If the next event is a class, the size of the audience is also sampled.

Figure 4 plots the generated congestion of two corridor cells during the first week of February 2021. On the top, the congestion of a corridor cell at the building's entrance is shown while on the bottom the congestion of an intersection cell is shown. Both cells follow a realistic distribution in the sense that during the night-time

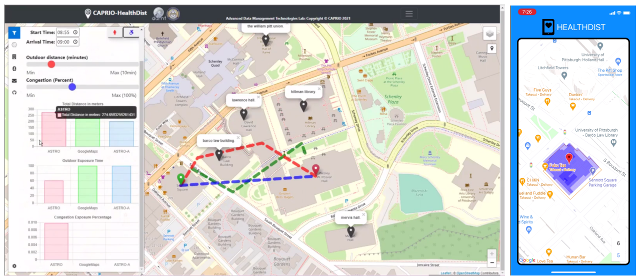


Figure 5: The *CAPRIO* system uses *EPICGen* under the hood to train accurate congestion forecasting models and recommend routes that reduce exposure risk by avoiding congested buildings.

only a very small number of individuals is observed whereas the number of individuals increases during the day and peaks around the beginning and ending of classes. As expected, the congestion at the intersection is much higher than at the entrance. This happens because individuals are very likely to walk by the intersection irrespective of the entrance or exit they used.

3 DEMONSTRATION DESCRIPTION

During the demonstration, the attendees will be able to comprehend the key concepts of *EPICGen*, the visualization abstraction, as well as the performance of our propositions by interacting with a user-friendly interface. Below we will present more details on the implementation of *EPICGen*, and then discuss our demo plan.

3.1 Demo Artifact

We have implemented *EPICGen* using an interactive map, integrating *Grid Overlayer* in the backend, which was developed using Play Framework 2.7[3]. The *EPICGen* web interface is implemented in HTML5/CSS3 along with extensive usage of Leaflet[4].

An illustrative congestion visualization interface is shown in Figure 1. We implement a query sidebar that allows the user to parameterize the components of *EPICGen*. Particularly, the query sidebar has two main tabs: (i) the *Grid Overlayer* parameters tab that enables the user to choose the building and grid size; and (ii) the *Congestion Generator* parameters tab that enables the user to tune the congestion generation. A submit button triggers the process of modeling the building and generating the congestion using the provided parameters.

The hardware stack of our *EPICGen* installation resides on a dedicated server. The server is featuring 12GB of RAM with 4 Cores (@ 2.90GHz). During the demonstration, we will connect over cable or Wi-Fi to the *EPICGen* web service and enable the users to interact with our intuitive web interface, as described next. We shall also have video recordings at hand, in case the network is unstable at the conference.

[3]Play Framework: https://www.playframework.com/
[4]Leaflet: https://leafletjs.com/

3.2 Demo Plan

Equipment: The conference attendees will have the opportunity to interactively engage with the *EPICGen* GUI using a standard laptop, a tablet and smartphones we will bring along at the conference.

Datasets: We will pre-load floor plans for a variety of buildings in the University of Pittsburgh campus to the *EPICGen* back-end.

Scenario 1: *EPICGen*'s server will be publicly available to allow attendees to experiment with the parameters of the system and to see the result in real time on the interface. We will provide visual cues that will enable the audience to understand the benefits of our propositions.

Scenario 2: Additionally, attendees will have the opportunity to interact with our prototype *HealthDist* [6, 9] MCAN application, which was built on top of our *CAPRIO* architecture [5]. We developed *EPICGen* to evaluate *HealthDist*'s accuracy of forecasting the congestion in buildings and its effectiveness of recommending paths that minimize the exposure risk to airborne viruses. Figure 5 shows the interface of *HealthDist/CAPRIO* comparing three paths. The first path (green color) is recommended by Google Maps and due to lack of indoor information it recommends an entirely outdoor path. The second path (blue color) is recommended by *HealthDist/CAPRIO* and is similar to the first path due to an accessibility constraint that was imposed by the user. The last path (red) is also recommended by *HealthDist/CAPRIO* with the difference that no accessibility constraints are imposed. *HealthDist/CAPRIO* paths leverage indoor information and congestion forecasting to recommend indoor paths that reduce the viral exposure risk by avoiding congested corridors.

ACKNOWLEDGMENTS

This work was partially funded by NIH award U01HL137159, NSF grant CNS-2027794, and by the Pittsburgh Foundation and reflects only the authors' opinions.

REFERENCES

[1] Soheila Abrishami and Piyush Kumar. 2018. Using real-world store data for foot traffic forecasting. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 1885–1890.
[2] Soheila Abrishami, Piyush Kumar, and Wickus Nienaber. 2017. Smart stores: A scalable foot traffic collection and prediction system. In *Industrial Conference on Data Mining*. Springer, 107–121.
[3] Vincent Bindschaedler and Reza Shokri. 2016. Synthesizing plausible privacy-preserving location traces. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 546–563.
[4] Karl Brierley. 2013. The effects of pedestrian delay and overcrowding on our streets and the rationale for shorter blocks and through blocks links. *A report prepared for the city of Melbourne* (2013).
[5] Constantinos Costa, Xiaoyu Ge, and Panos K. Chrysanthis. 2019. CAPRIO: Graph-based Integration of Indoor and Outdoor Data for Path Discovery. *Proc. VLDB Endow.* 12, 12 (2019), 1878–1881.
[6] Constantinos Costa, Brian T. Nixon, Sayantani Bhattacharjee, Benjamin Graybill, Demetrios Zeinalipour-Yazti, Walter Schneider, and Panos K. Chrysanthis. 2021. A Context, Location and Preference-Aware System for Safe Pedestrian Mobility. In *22nd IEEE International Conference on Mobile Data Management (MDM)*. 217–224.
[7] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3, 1 (2013), 1–5.
[8] Huan Li, Hua Lu, Xin Chen, Gang Chen, Ke Chen, and Lidan Shou. 2016. Vita: A Versatile Toolkit for Generating Indoor Mobility Data for Real-World Buildings. *Proc. VLDB Endow.* 9, 13 (2016), 1453–1456.
[9] Brian T. Nixon, Sayantani Bhattacharjee, Benjamin Graybill, Constantinos Costa, Sudhir Pathak, Walter Schneider, and Panos K. Chrysanthis. 2021. HealthDist: A Context, Location and Preference-Aware System for Safe Navigation. In *22nd IEEE International Conference on Mobile Data Management (MDM)*. 250–253.