Frontmatter

Editors

SIGAPP FY’18 Quarterly Report

J. Hong

SAC 2018 Preview

H. Haddad

Selected Research Articles

Design and Evaluation of a Privacy Architecture for Crowdsensing Applications

A. Perez and S. Zeadally

A Pipeline-Based Heterogeneous Framework for Efficient Synthetic Light Field Rendering

C. Kao, L. Tseng, and W. Hsu

Applying Software-Defined Networking to Minimize the End-to-End Delay of Network Services

T. Chin, M. Rahouti, and K. Xiong

NMF Based Sparse Cholesky Decomposition Technique for Dimensionality Scale Back in Large Data Sets

J. Alostad
Applied Computing Review

Editor in Chief
Sung Y. Shin

Associate Editors
Hisham Haddad
Jiman Hong
John Kim
Tei-Wei Kuo
Maria Lencastre

Editorial Board Members

Sheikh Iqbal Ahamed
Gail-Joon Ahn
Davide Ancona
Massimo Bartoletti
Thais Vasconcelos Batista
Maurice Ter Beek
Giampaolo Bella
Paolo Bellavista
Albert Bifet
Stefano Bistarelli
Gloria Bordogna
Marco Brambilla
Barrett Bryant
Antonio Bucchiarone
Andre Carvalho
Jaelson Castro
Martine Ceberio
Tomas Cerny
Priya Chandran
Li Chen
Seong-je Cho
Ilyoung Chong
Soon Chun
Juan Manuel Corchado
Luís Cruz-Filipe
Marília Curado
Fabiano Dalpiaz
Lieven Desmet
Mauro Dragoni
Khalil Drira
Ayman El-Baz
Ylies Falcone
Mario Freire
Joao Gama
Marisol García-Valls
Karl M. Goeschka

Aniruddha Gokhale
George Hamer
Hyoil Han
Ramzi Haraty
Mohammed El Hassouni
Jun Huang
Yin-Fu Huang
Angelo Di Iorio
Hasan Jamil
JoonHyouk Jang
Rachid Jennane
Jinman Jung
Soon Ki Jung
Sungwon Kang
Bongjae Kim
Dongkyun Kim
Sang-Wook Kim
Stefan Kramer
Daniel Kudenko
S.D Madhu Kumar
Tei-Wei Kuo
Paola Lecca
Byungjeong Lee
Jaeheung Lee
Axel Legay
Hong Va Leong
Frederic Loulergue
Tim A. Majchrzak
Cristian Mateos
Herman Melgratti
Mercedes Merayo
Marjan Mernik
Hong Min
Raffaela Miranda
Eric Monfroy
Marco Di Natale

Rui Oliveira
Flavio Oquendo
Ai-Chun Pang
Apostolos Papadopoulos
Gabriella Pasi
Anand Paul
Manuela Pereira
Ronald Petric
Peter Pietzuch
Ganesh Kumar Pugalendi
Rui P. Rocha
Pedro Pereira Rodrigues
Agostinho Rosa
Marco Rospocher
Davide Rossi
Giovanni Russello
Francesco Santini
Patrizia Scandurra
Jean-Marc Seigneur
Shaojie Shen
Eunjee Song
Junping Sun
Sangssoo Sung
Francesco Tiezzi
Dan Tulpan
Teresa Vazão
Mirko Virolí
Wei Wang
Raymond Wong
Jason Xue
Markus Zanker
Tao Zhang
Jun Zheng
Yong Zheng
SIGAPP FY’18 Quarterly Report

January 2018 – March 2018
Jiman Hong

Mission

To further the interests of the computing professionals engaged in the development of new computing applications and to transfer the capabilities of computing technology to new problem domains.

Officers

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>Jiman Hong</td>
<td>Soongsil University, South Korea</td>
</tr>
<tr>
<td>Vice Chair</td>
<td>Tei-Wei Kuo</td>
<td>National Taiwan University, Taiwan</td>
</tr>
<tr>
<td>Secretary</td>
<td>Maria Lencastre</td>
<td>University of Pernambuco, Brazil</td>
</tr>
<tr>
<td>Treasurer</td>
<td>John Kim</td>
<td>Utica College, USA</td>
</tr>
<tr>
<td>Webmaster</td>
<td>Hisham Haddad</td>
<td>Kennesaw State University, USA</td>
</tr>
<tr>
<td>Program Coordinator</td>
<td>Irene Frawley</td>
<td>ACM HQ, USA</td>
</tr>
</tbody>
</table>

Notice to Contributing Authors

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library and in any Digital Library related services
- to allow users to make a personal copy of the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will refer requests for republication directly to you.
SAC 2018 Preview

The 33rd Annual ACM Symposium on Applied Computing (SAC) will be held in Pau, France, Monday April 9 to Friday April 13, 2018, at the Historic Conference Center Palais Beaumont in downtown Pau. The Tutorials Program is planned for Monday; the Technical Program for Tuesday through Friday; the Student Research Competition (SRC) Program for Tuesday (display session) and Thursday (presentations session), respectively; and the Posters Program for Wednesday.

SAC 2018 has received 931 submissions, from 56 countries. To date, it is expected to have approximately 235 papers presented at SAC. The review process and acceptance decision resulted in an acceptance rate of 25% across all 40 tracks. In addition, 53 posters have been invited for participation in the Posters Program. These are papers that have gone through the review process as papers. The SRC Program received 51 submissions. After the review process by the respected track committees, 19 abstracts have been invited to compete during the SRC Program. The accepted abstracts will compete for three cash prizes ($500, $300, and $200) and winners will be recognized during the conference banquet event on Thursday April 12, 2018. The first place winner proceeds to the National ACM SRC program. Furthermore, 11 tutorial proposals were reviewed by the organizing committee and 5 tutorials have been invited to participate in the Tutorials Program. Tutorials Program details are posted on the conference website.

<table>
<thead>
<tr>
<th>#</th>
<th>Track</th>
<th>Submissions</th>
<th>Accepted Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACMMPH</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>BIO</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>BPMEA</td>
<td>28</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>CC</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>CCS</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>CIVIA</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>CoCo</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>CPS</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>DADS</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>DM</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>DS</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>DTTA</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>EMBS</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>HCI</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>IAR</td>
<td>21</td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td>IoT</td>
<td>35</td>
<td>9</td>
</tr>
<tr>
<td>17</td>
<td>IRMAS</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>KEGeoD</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>KRR</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>MCA</td>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>NET</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>22</td>
<td>OOPPS</td>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>23</td>
<td>OS</td>
<td>32</td>
<td>11</td>
</tr>
<tr>
<td>24</td>
<td>PDP</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>PL</td>
<td>24</td>
<td>9</td>
</tr>
<tr>
<td>26</td>
<td>RE</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>27</td>
<td>RS</td>
<td>37</td>
<td>10</td>
</tr>
<tr>
<td>28</td>
<td>SATTA</td>
<td>23</td>
<td>9</td>
</tr>
<tr>
<td>29</td>
<td>SE</td>
<td>55</td>
<td>14</td>
</tr>
<tr>
<td>30</td>
<td>SEC</td>
<td>35</td>
<td>8</td>
</tr>
<tr>
<td>31</td>
<td>SiSoS</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>32</td>
<td>SOAP</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>33</td>
<td>SONAMA</td>
<td>41</td>
<td>13</td>
</tr>
<tr>
<td>34</td>
<td>SP</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>35</td>
<td>SVT</td>
<td>43</td>
<td>11</td>
</tr>
<tr>
<td>36</td>
<td>SWA</td>
<td>25</td>
<td>7</td>
</tr>
<tr>
<td>37</td>
<td>UE</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>38</td>
<td>VSPLE</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>39</td>
<td>WCN</td>
<td>27</td>
<td>9</td>
</tr>
<tr>
<td>40</td>
<td>WT</td>
<td>23</td>
<td>7</td>
</tr>
</tbody>
</table>

Information about hotels, transportation, and excursions is posted on the conference website (https://www.sigapp.org/sac/sac2018/). The conference registration includes daily lunches, coffee breaks, a reception on Tuesday, and a banquet dinner on Thursday. The banquet dinner will be held in the costal city of Biarritz, at the city congress center (http://congres.biarritz.fr/en).
<table>
<thead>
<tr>
<th>#</th>
<th>Country</th>
<th>Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Algeria</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Australia</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Austria</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Belgium</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Brazil</td>
<td>206</td>
</tr>
<tr>
<td>6</td>
<td>Bulgaria</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Canada</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Chile</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>China</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>Czech Republic</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Denmark</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>Egypt</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>Estonia</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Finland</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>France</td>
<td>83</td>
</tr>
<tr>
<td>16</td>
<td>Germany</td>
<td>56</td>
</tr>
<tr>
<td>17</td>
<td>Greece</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>Iceland</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>India</td>
<td>38</td>
</tr>
<tr>
<td>20</td>
<td>Iran</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>Ireland</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>Israel</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>Italy</td>
<td>44</td>
</tr>
<tr>
<td>24</td>
<td>Japan</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>Kuwait</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>Lebanon</td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td>Lithuania</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>Algeria</td>
<td>10</td>
</tr>
</tbody>
</table>

The Steering and Organizing committees are pleased to have SAC 2018 in the city of Pau, France. We invite you to join us this April to meet other attendees, enjoy the conference programs, and have a pleasant stay in Pau. We hope to see you there.

On Behalf of SAC Steering Committee,

Hisham M. Haddad
Conference Co-Chair and Member of the Steering Committee

Next Issue

The planned release for the next issue of ACR is June 2018.
Design and Evaluation of a Privacy Architecture for Crowdsensing Applications

Alfredo J. Perez
Columbus State University
4225 University Ave. CCT 424
Columbus, GA, USA 31907
perez_alfredo@columbusstate.edu

Sherali Zeadally
University of Kentucky
315 Little Library Building
Lexington, KY, USA 40506-0224
szeadally@uky.edu

ABSTRACT
By using consumer devices such as cellphones, wearables and Internet of Things devices owned by citizens, crowdsensing systems are providing solutions to the community in areas such as transportation, security, entertainment and the environment through the collection of various types of sensor data. Privacy is a major issue in these systems because the data collected can potentially reveal aspects considered private by the contributors of data. We propose the Privacy-Enabled ARchitecture (PEAR), a layered architecture aimed at protecting privacy in privacy-aware crowdsensing systems. We identify and describe the layers of the architecture. We propose and evaluate the design of MetroTrack, a crowdsensing system that is based on the proposed PEAR architecture.

CCS Concepts
• Networks ➝ Layering; Sensor Networks; • Security and Privacy ➝ Security Services; • Human-centered computing ➝ Ubiquitous and mobile computing;

Keywords
Privacy; Security; Crowdsensing; Community-based Sensing; Participatory Sensing; Opportunistic Sensing; Internet of Things; Wireless Sensor Networks.

1. INTRODUCTION
The ubiquity of Internet-connected consumer devices have the potential to improve issues affecting communities through the crowdsensing paradigm. In crowdsensing systems (also known as community-based sensing systems), people collect sensor data using their own consumer devices such as cellphones, wearables, and Internet of Things (IoT) devices (e.g., Internet-connected cars) with the goal of helping a community (e.g., a city or town) in areas such as environmental monitoring, transportation, entertainment, security, and healthcare.

Recent technological advances in the last decade in the miniaturization of sensors, computing power, and the mobile Internet, along with the ubiquity of these consumer devices have enabled the implementation and deployment crowdsensing systems [21][30]. The protection of privacy in crowdsensing systems plays an important role in their successful development and deployment because the use of consumer devices to collect sensor data presents significant privacy risks to all of the participants involved. Thus, the use of Internet-connected consumer devices in crowdsensing faces a tradeoff: on one hand we need to collect data as accurately as possible, but on the other hand the collection and sharing of crowdsensing data must preserve the privacy of users [1][31].

We review the privacy issues and solutions in crowdsensing systems, then we present the Privacy-Enabled Architecture (PEAR), a layered architecture aimed at protecting the privacy of users involved in crowdsensing systems. We describe the main components of the architecture, and we evaluate MetroTrack, an implementation of the proposed PEAR architecture.

2. CROWDSensing SYSTEMS
Advances and integration of Micro Electro-Mechanical Systems (MEMS) with communication and computing devices in the 1990’s paved the way for the development of Wireless Sensor Networks (WSNs). Even though early research in this area was performed in the 1970’s through the Defense Advanced Research Project Agency’s (DARPA) Distributed Sensor Networks program [30], the vision behind the development of WSNs during the 1990’s was to collect data of interest in a particular geographical area while minimizing the cost to deploy and maintain the network, and maximizing the lifetime of the WSN and the coverage area the WSN monitors. Thus, the devices that were developed for WSNs were of small sizes (as small as 2 mm²) while at the same time these devices could sense, transmit data, and harvest their energy needs from the environment. It was envisioned that WSNs could be active for years without human intervention and to be cheap enough to be deployed everywhere to enable the deployment of thousands of sensing devices in a single WSN [30].

As research progressed in the development of WSN systems, certain goals, in particular the ones that minimize the costs of deployment and maintenance were not achievable when deploying thousands of devices in a single WSN as initially thought. The consequence was that the actual deployments of static WSNs were done as small and focused WSNs deployments with only a few hundreds of devices [30]. In the early 21st century, cellular devices started to become ubiquitous and, in the past decade, these cellular devices became powerful enough to connect to the Internet. Eventually, these mobile devices were manufactured with integrated sensors such as accelerometers, gyroscopes, location sensors which could be potentially used to collect different types of data of interest.

Crowdsensing systems have been developed during last decade as an alternative to static wireless sensor networks in urban environments to address the collection of data in an inexpensive manner. These systems take advantage of the availability of the mobile Internet and the ubiquity of powerful, sensor-enabled consumer devices such as mobile phones, wearables and IoT devices [5][30]. Application areas where these systems are being deployed include environmental monitoring, transportation, entertainment, security, among others [17][20][21][22][27][28]. In this section we provide an overview of crowdsensing systems, the threats to privacy in crowdsensing systems, and a review of the solutions available to help mitigate these threats.
2.1 Hardware Architectures and Actors

The typical hardware components of crowdsensing systems are presented in figure 1, and they include [30]:

- **Sensors**: These components collect data (e.g., temperature, movement, noise, images) from physical actions or processes.
- **First-level integrators**: They perform initial data verification, aggregation and basic analysis (e.g., feature extraction) on the data collected by sensors.
- **Data transport**: In crowdsensing systems, data transport is provided by the Internet or any communication network that enables the end-to-end transfer.
- **Second-level integrators**: These components collect and store data sent from first-level integrators. They also provide analytics services and feedback to first-level integrator devices and to external entities.

These components are utilized to collect sensor data through **sensing tasks** which are software applications or scripts executed at first-level integrator devices. There are three major actors that participate in the data collection for crowdsensing systems:

- **Task organizers**: They develop the sensing tasks that will be deployed at first-level integrators.
- **Participants**: They represent the entities that own or are in custody of first-level integrator devices. They may be provided feedback based on the collected data.
- **External entities**: They represent third parties whom task organizers can share data (or information) generated from the sensor data.

The actors interact with the hardware in the crowdsensing system with the goal to collect sensor data and metadata through the data collection cycle shown in figure 2. This collection cycle is composed of the following processes:

- **Task distribution**: The goal of task distribution is to release the sensing task to user participants. This is accomplished in two ways: participants either makes use of the sensing task from a server (second-level integrators), or the task is pushed to the users’ devices (first-level integrators) from second-level integrators.
- **Data collection**: Once tasks are configured at the participants’ devices, the tasks perform sensing and initial analysis that may include extracting features from sensor data, smoothing and filtering of outliers in the data, and data analytics that can be performed locally without the need of second-level integrators.
- **Data submission**: Tasks that execute at first-level integrators forward the collected data to second-level integrator devices. Depending on the system’s goals, data submission can be performed continuously or based on events (identified in the data collection process), and data can be submitted in real time or later (e.g., at the end of the day).
- **Data analysis and sharing**: In this process, second-level integrator devices use the collected data from first-level
integrators to perform analytics services (e.g., data analysis and machine learning) and provide feedback to first-level integrator devices. The feedback may include the release of new sensing tasks to user participants, resulting in a new data collection cycle. In addition, data may be released to external parties outside the system through this process.

3. RELATED WORKS ON PRIVACY SOLUTIONS FOR CROWDSENSING

As the devices that collect sensor data in crowdsensing systems are usually owned by citizens, the protection of their privacy becomes an important issue for the successful deployment of these systems. The data collected could be potentially linked with the identities of participants or the data could reveal aspects about individuals that are considered private [1][6][31][32][36]. Attacks to privacy in crowdsensing may be broadly classified as (1) re-identification attacks; (2) contextual attacks. We summarize the advantages and disadvantages of privacy protection mechanisms to mitigate re-identification and contextual attacks in table 1 and table 2 respectively.

3.1 Re-identification Attacks

Re-identification attacks are successful when a rogue entity discovers the identity of participants from the data (or metadata) collected (or submitted by the participants) in the system. These attacks may occur because the participants’ identities are inferred from metadata such as network addresses/identifiers (e.g., IP addresses, MAC addresses and cookies) which are needed by network protocols to send and receive data, or they may be accomplished through the discovery of identities from any of the tasks performed by the crowdsensing system to collect data (figure 2). Privacy protection mechanisms against re-identification attacks which use network identifiers can be achieved by using double encryption via brokers [36][38], peer-to-peer (P2P) anonymization networks [2][6], and the utilization of disposable network identifiers such as pseudonyms [8][10][13][35].

In the case of re-identification attacks, because of the management of sensing tasks in the crowdsensing system, privacy solutions depend on the processes involved during data collection (as shown in figure 2). For example, for task distribution, the privacy solutions to avoid re-identification include the utilization of beacons that distribute tasks through the broadcasting of signals from the beacons or access points (such as WiFi access points) to the first-level integrator devices [36], task downloading at crowded spaces (in the case of mobile, first-level integrator devices) [36], and the use of anonymization networks [2]. In the case of data collection and submission processes, solutions to handle re-identification include the use of group-based signatures [15], data aggregation [34], and the use of representative samples from the data collected in a region [16][38].

3.2 Contextual Attacks

Given the data (or metadata) submitted to the system by participants, contextual attacks attempt to discover and associate aspects considered private by participants with their identity. Examples in this category include inferring contexts such as places, activities, behaviors, and/or health state based on the collected data. The goal of these solutions is to diminish or eliminate the risks of discovering and associating aspects or contexts considered private by participants from the data (or metadata) submitted to the system. Table 2 presents the advantages and disadvantages of privacy protection mechanisms to mitigate re-identification attacks.

To thwart contextual attacks, privacy solutions can be divided into two groups: (1) solutions to manage contextual attacks at the data collection stage; (2) solutions to manage contextual attacks when the collected data is shared externally (to third parties). In the first group of solutions (privacy protection at data collection) solutions include the bubble sensing [24] approach which allows data collection in a particular context and the virtual walls approach which denies data collection at predefined contexts [7]. In the case of external data sharing, solutions to handle contextual attacks include anonymization in the release of microdata (e.g., k-anonymity [37], l-diversity [25], and t-closeness [23]), and methods to release aggregated data (e.g., statistical data) [9][12].

We summarize the main research contributions of this work as follows:

- We propose the design of a privacy architecture called Privacy Enabled Architecture (PEAR) which is a layered software architecture for crowdsensing systems wherein first-level integrators and second-level integrators are abstracted as data contributors. This allows second-level integrators to become data providers for other crowdsensing systems without defining new layers.
- We present the implementation of MetroTrack, a crowdsensing system based on the PEAR architecture that runs on the JavaEE frameworks and the Android platform.
- We evaluate the privacy mechanisms incorporated in the design of MetroTrack to protect participants’ privacy.

4. THE PRIVACY ENABLED ARCHITECTURE (PEAR)

We propose the PEAR architecture which consists of four abstraction layers namely communication, anonymization, security and privacy, and processing. First-level and second-level integrator devices are referred to as integrators in the PEAR architecture, as this abstraction allows second-level integrator devices to become data providers (i.e., first-level integrators) for other crowdsensing systems without having to define new layers to share data. This architecture also manages scalability by creating networks of second-level integrator devices. Figure 3 depicts an example of a PEAR-enabled system.

4.1 Communication Layer

This layer abstracts the communication protocols between integrator devices. Usually these protocols are implemented by the TCP/IP protocol stack suite. However, given the ubiquity of cellular networks in crowdsensing systems, communication protocols for crowdsensing could be implemented over the Multimedia Messaging System (MMS) infrastructure, or Short Messaging System (SMS).

4.2 Anonymization Layer

This layer implements mechanisms that allow integrator devices to hide network location identifiers (e.g., IP addresses) from other integrators and external parties to avoid re-identification attacks. These anonymization mechanisms may be implemented through trusted third parties [38], or by using peer-to-peer anonymization networks (e.g., Tor). Systems may bypass anonymization depending on the goals of the system, or if participants give consent to include network identifiers as part of the data collected by the crowdsensing system.
<table>
<thead>
<tr>
<th>Type of attack</th>
<th>Protection mechanism</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re-identification attacks from network identifiers</td>
<td>Double encryption via brokers [36][38]</td>
<td>Easy to implement</td>
<td>Requires a trusted third party to forward the data before submission to cloud services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protects participants’ IP network identifiers</td>
<td>May have significant power overhead due to encryption, an issue for battery-powered devices</td>
</tr>
<tr>
<td>Peer-to-Peer (P2P) anonymization networks [2][6]</td>
<td>Does not require a trusted third-party</td>
<td></td>
<td>May not be available to all crowdsensing devices due to OS or hardware constraints</td>
</tr>
<tr>
<td></td>
<td>Protects participants’ IP network identifiers</td>
<td></td>
<td>May have significant power overhead due to P2P network maintenance, which is an issue for battery-powered devices</td>
</tr>
<tr>
<td>Disposable network identifiers [8][10][13][35]</td>
<td>Lightweight</td>
<td>Mechanism can be implemented on any crowdsensing device</td>
<td>May not protect participants from some type of network re-identification (e.g., IP addresses)</td>
</tr>
<tr>
<td>Broadcasting of signals from beacons to download sensing tasks [36]</td>
<td>Hides identity of participant by not having the participant’s device to initiate network communication (device only receives data)</td>
<td></td>
<td>May require the modification of the infrastructure (e.g., WiFi access points) or special agreements to broadcast signals</td>
</tr>
<tr>
<td>Task downloading at crowded spaces [36]</td>
<td>Assumes that there are enough people at a given place so that the identities of the participants are hidden in the crowd</td>
<td></td>
<td>May not be possible to use in rural or isolated places due to the difficulty in getting a crowd</td>
</tr>
<tr>
<td>Use of Anonymization networks for task downloading [2]</td>
<td>Hides participant’s identity through by forwarding the data to the anonymization system</td>
<td></td>
<td>May not be available to all crowdsensing devices due to OS or hardware constraints</td>
</tr>
<tr>
<td>Group-based signatures for data submission [15]</td>
<td>May be easy to implement as it may be implemented as a shared credential</td>
<td></td>
<td>Privacy protection depends on the size of the group to guarantee low probability of re-identification</td>
</tr>
<tr>
<td>Data aggregation [34]</td>
<td>Conceals identity by aggregating data (e.g., calculating averages) on participants’ values before submission</td>
<td></td>
<td>Requires coordination between participants to calculate aggregates</td>
</tr>
<tr>
<td>Use of representative samples of data collected in a region [16][38]</td>
<td>Conceals identity by choosing best/representative samples from a pool of participants</td>
<td></td>
<td>Require coordination between participant’s devices to select the representative samples</td>
</tr>
<tr>
<td>Type of attack</td>
<td>Protection mechanism</td>
<td>Advantages</td>
<td>Disadvantages</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Contextual attacks during data collection</td>
<td>Method allows data collection at specific contexts [24]</td>
<td>Allows participant to specify rules to collect data, giving them more autonomy where to collect</td>
<td>Contexts may be hard to specify and may require also Artificial Intelligence (AI) to create usable rules</td>
</tr>
<tr>
<td></td>
<td>Method denies data collection at specific contexts [7]</td>
<td>Allows participant to specify rules to collect data, giving them more autonomy where to collect</td>
<td>Contexts may be hard to specify and may require also AI to create usable rules</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Participants may inadvertently leak private data as a result of not specifying contexts correctly</td>
</tr>
<tr>
<td>Contextual attacks when data is shared externally</td>
<td>k-anonymity [37]</td>
<td>Works on microdata release</td>
<td>k-anonymity does not protect against attacks when background knowledge is used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protects against identity disclosure</td>
<td>Does not protect privacy if all the values in the k-anonymity group share the same sensitive attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simple to implement and understand because any record in the microdata has at least k-1 records with similar attributes to hide the real record(s) to be protected</td>
<td></td>
</tr>
<tr>
<td></td>
<td>l-diversity [25]</td>
<td>Works on microdata release</td>
<td>May not protect the microdata release from attacks such as the skewness attack (the distribution of the sensitive values in the anonymization group is skewed from the overall distribution of the microdata) and the similarity attack in which all the values in the anonymization group for the records in the microdata have the same semantic (real life) meaning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protects against sensitive attributes’ disclosures</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Extends k-anonymity by imposing the condition that well-represented values must be present in the group of data to be anonymized</td>
<td></td>
</tr>
<tr>
<td></td>
<td>t-closeness [23]</td>
<td>Works on microdata release</td>
<td>The value t may be hard to determine depending on data to be protected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protects against skewness and similarity attacks (improves over l-diversity)</td>
<td>The t value may have different privacy implications depending on the type of data (e.g., a particular t value can be interpreted differently for numeric and/or categorical data)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assures that the distribution on the sensitive values in the anonymization group has a similar distribution when compared with the overall distribution of the sensitive values in the microdata</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Differential privacy [9][12]</td>
<td>Works on statistical (summarized) data release</td>
<td>The calculation of the statistical value to achieve differential privacy depends on the type of data used to calculate the statistical value (e.g., it differs between categorical and numerical data)</td>
</tr>
</tbody>
</table>
4.3 Security and Privacy Layer
This layer implements mechanisms and protocols to encrypt data between integrator devices, and includes privacy-preserving mechanisms for integrator devices. Security mechanisms in this layer include symmetric and asymmetric cryptographic methods and protocols that guarantee end-to-end security between integrator devices. Privacy mechanisms in this layer implement algorithms/procedures to allow participants to handle their exposure to context privacy attacks (e.g., privacy rules, algorithms to handle location privacy) and mechanisms to handle privacy for second-level integrators when sharing bulk data release (microdata release) [9] and aggregated (summarized, statistical) data release with external parties.

4.4 Processing Layer
This layer includes mechanisms and protocols that collect and analyze sensor data. These mechanisms may be implemented at first-level integrator devices to perform initial data analysis (e.g., feature extraction, data smoothing) and at second-level integrators to extract information (e.g., outlier detection, machine learning). The processing layer may include mechanisms to handle privacy implemented by task organizers for integrator devices (e.g., a task organizer specifies where sensor data should not be collected) and may also include mechanisms to provide incentives for participants. The processing layer mechanisms make use of software Application Programming Interfaces (e.g., an app development framework, a server-side framework) for their implementation. The processing layer also implements mechanisms to perform efficient data collection (e.g., power optimization in case of first-level integrator devices).

5. MetroTrack: A PROTOTYPE SYSTEM USING PEAR
We describe the design of a prototype implementation of the PEAR architecture called MetroTrack, a system on which a city administration (task manager) can issue crowdsensing tasks to its citizens (participants) to collect data of interest. In this system, the citizens participate altruistically in the data collection, and tasks can be either participatory (e.g., uploading of photos/videos for security [3]) or opportunistic (e.g., tracking of road congestion, or road maintenance status [26]). MetroTrack consists of client (participant) applications executing on Android-enabled devices, and server-side components that are deployed in the cloud using the Java Enterprise Edition (EE) framework.
5.1 MetroTrack’s Mobile Components

Four applications make up the MetroTrack’s mobile components for the Android OS (figure 4). Each of these components implements different layers of the proposed PEAR architectural model.

5.1.1 Orbot client

MetroTrack makes use of the Orbot [29] application which is the Tor’s network proxy for Android. The Tor network [11] is an anonymization network that provides network anonymization to TCP flows. Tor works by having the client to create a path through Tor hosts from the client to a server. Messages along this path are encapsulated into layers of encryption (like an onion) at the client, and each host along the path removes an encryption layer (like peeling an onion), which allows the current host to know the next host to forward the message. Once the final layer is decrypted, the last host delivers the message to its original destination [11].

Android applications can use Orbot to access a server in the Internet through a local proxy in the device, or can incorporate Orbot as a component within an application. In our current design, Orbot is used as a proxy. Orbot is open source and can be downloaded from the Google Play market. Orbot serves as part of the anonymization and communication layers of the PEAR architecture for MetroTrack. A screenshot of the Orbot client is shown in figure 5.

5.1.2 MetroTrackTaskAgent

The MetroTrackTaskAgent is used by the MetroTrack system to deliver participants’ information about new tasks issued in the system. The agent can also notify participants about updates on previously issued tasks. As shown in figure 4, MetroTrackTaskAgent uses Orbot to connect to the server components of MetroTrack.

The agent retrieves notifications about new tasks available to participants from the MetroTrack server components. Since Orbot does not provide end-to-end security, the task agent must secure the requests before using Orbot. Transport Layer Security (TLS) security provides the security mechanism for the task agent. The MetroTaskAgent is part of the processing layer of PEAR, with the security components of the task agent being part of the security and privacy layer of PEAR. Participants can download the task agent from the Google Play market.

5.1.3 UserPrivacyManager

Through this component, participants can configure their own privacy settings for SensingTasks. These privacy settings are implemented as privacy rules based on sensor and date/time data, and they can be implemented as simple rules (e.g., “don’t provide data to this task if close to a particular location”), or more complex contextual rules based on activity recognition (e.g., “don’t provide data if sleeping”). This module is composed of subcomponents such as PrivacyPolicyManager which implements the rules, the ActivityRecognizer module which recognizes activities based on sensor data, and the PrivacyContentProvider which provides sensor data to tasks based on the decisions of the PrivacyPolicyManager. The SensorManager and LocationManager components are part of the Android API and provide information to the PrivacyPolicyManager subcomponent.

The PrivacyContentProvider subcomponent provides raw data to a sensing task based on the decision of the privacy rules. In the current design, the UserPrivacyManager could be downloaded from Google Play. However, this module could become part of mobile operating systems as part of the privacy/security settings.

5.1.4 SensingTask

This component implements the collection of data for the task manager. Sensing tasks are downloaded from MetroTrack’s servers using Orbot. Each download has a unique identifier that is hardcoded when the SensingTask is compiled as an app ready to install. This design allows MetroTrack servers to authenticate the each of the task installations instead of authenticating participants. The rationale is that MetroTrack only needs to make sure that the data is coming from an authorized party, and this can be accomplished by hardcoding IDs into each download of the app.

Figure 5. Orbot Android client.

Figure 6. PhotoPriv: A MetroTrack task.
Subcomponents of this module include: the DataCollectionModule which collects data from the UserPrivacyManager module and performs basic data analysis (e.g., feature extraction), the TaskPrivacyManager which implements privacy rules established by the task manager (and also includes mechanisms to show consents to participants), the DataSubmissionModule which prepares the data for submission, and the SecurityManager which manages authentication, session establishment, and end-to-end encryption with the server. The SecurityManager may use the MetroTrackTaskAgent to check if the current task is still valid. The SecurityManager also utilizes the Orbot component to submit data to the MetroTrackServers.

Figure 6 shows an example of a MetroTrack SensingTask called PhotoPriv. In this task, citizens are notified when people or objects of interest (e.g., stolen cars, hit and runs, amber alerts, thieves) are present in a location where a law enforcement agency may be attempting to locate by using crowdsensing. Contributors can use PhotoPriv to send anonymous geo-located photos that are uploaded with a message to law enforcement agencies. We used the NetCipher library in PhotoPriv to send data through the Orbot app to the server.

5.2 MetroTrack’s Server Components
The MetroTrack’s server consists of four major components, namely SensingTaskManager, DataStorage, DataAnalysis, and ExternalPrivacyManager. Figure 6 shows the flow of data among these components. Our current design assumes that these components will execute in a Java EE application server (e.g., Glassfish server).

5.2.1 SensingTaskManager
This component handles the management of sensing tasks for MetroTrack’s mobile components. The SensingTaskManager is used by the task organizer to announce new SensingTask apps to participants’ devices. As mentioned in the previous section, each instance of a SensingTask downloaded by participants has its own identifier which allows it to be authenticated by MetroTrack servers. This is performed by having different compilations of the same SensingTask and offering these on demand. SensingTasks are meant to be lightweight, and a background process in the SensingTaskManager is constantly compiling and caching the SensingTasks. The SensingTaskManager component also provides the mechanisms to handle security, authentication and sets up privacy rules for SensingTasks.

5.2.2 DataStorage
This component abstracts the operations needed to store the data received by the SensingTaskManager into database systems. Depending of the type of SensingTask, the data may be structured, unstructured, or a combination of both types of data. PostgreSQL, MySQL and other database management systems (or regular file systems) may be used to store data.

5.2.3 DataAnalysis
This component allows a task manager to perform inference, correlation, and data analysis based on the data received from SensingTasks. This component can filter outliers, detect trends and patterns, and perform data analysis that could be only performed at the server. This module allows a task organizer to have a complete picture of the situation being studied. Task organizers may take measures such as preparing and releasing new tasks, or providing reports to third parties.

5.2.4 ExternalPrivacyManager
MetroTrack makes use of this component to handle privacy when data is shared with external systems or parties. The algorithms implemented in this module include mechanisms such as k-anonymity [37], l-diversity [25], t-closeness [23] to handle privacy for bulk data release (microdata release). For aggregated data, differential privacy mechanisms [12] may be used.

5.3 Analysis of MetroTrack
5.3.1 Evaluation of privacy goals in MetroTrack
MetroTrack system was designed to mitigate re-identification and contextual privacy attacks described in section 2.2. It achieves this by implementing privacy protection solutions through independent components at the mobile devices and the server which isolate the access to sensor data and provide the means for users to handle identifiable data. We summarize how the components in MetroTrack mitigate the various attacks in table 3.

5.3.2 Tradeoff between privacy and estimation
Allowing participants to establish their own privacy rules may induce noise in the estimation performed by second-level integrators from first-level integrators’ sensor data. More research is needed to investigate the tradeoff between the participant’s privacy rules versus the information loss in the system [38].
5.3.3 Tor as an anonymization network

The utilization of Tor as the anonymization layer in MetroTrack does not allow to perform UDP transmission because Tor supports only TCP flows. UDP may be needed when a sensing task needs to deliver real-time sensor data to second-level integrators. As such, the evaluation of alternative solutions for providing anonymous network transfers from the point of view of privacy protection, quality of service and power consumption are needed.

5.3.4 Layered architectural issues

A layered architecture, as utilized in MetroTrack, may consume more power at first-level integrator devices than a cross-layer design. Static wireless sensor network research has shown reduced power consumption of cross layer designs over layered designs. One possible solution to improve power consumption and at the same time enforce privacy is by using cloudlets [33] which are software modules that can be deployed in virtual machines in the cloud to offload processing from a mobile device.

5.3.5 Choice of authentication mechanism

The proposed authentication method requires multiple compilations for the same task because each of them will have its own hard-coded identification code. As such, the server may need additional storage as well as some type of background processing to keep enough compiled tasks available. In late 2016, it was found that the average Android app size is 15 MB [4]. Using 1TB SSD hard drive dedicated for this purpose could hold more than 6 million of these tasks.

Assuming that it takes one minute to compile an Android app and that task organizers use only a single machine with only one core to compile for 24 hours, 1440 tasks could be compiled per day. Suppose that the task organizer uses a computer with 10 cores and enough RAM to compile tasks simultaneously, up to 10,000 tasks could be compiled per day. To deploy a sensing task that could be used by every resident in the New York metropolitan area (~20 million according to the 2015 US Census), a task organizer would need 100 machines working for 2 full days to generate enough sensing tasks for each inhabitant, which is feasible.

6. CONCLUSION

There has been a growing interest in the development of privacy-preserving architectures for crowdsensing systems in the last few years. To handle privacy issues when developing crowdsensing systems, this work has proposed the PEAR architecture. We described the components of the architecture and we presented a pro-

### Table 3. MetroTrack’s privacy mechanisms.

<table>
<thead>
<tr>
<th>Data collection process</th>
<th>Privacy threat</th>
<th>MetroTrack privacy mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task distribution</td>
<td>Re-identification attacks</td>
<td>• Use of Orbot client to hide network identifiers when a SensingTask is downloaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use of the SecurityManager in both the MetroTrackTaskAgent and in the SensingTask to encrypt data between mobile client and server</td>
</tr>
<tr>
<td>Data collection</td>
<td>Re-identification attacks</td>
<td>• Use of the TaskPrivacyManager in the SensingTask module to implement data privacy algorithms (e.g., k-anonymity, l-diversity)</td>
</tr>
<tr>
<td></td>
<td>Contextual attacks</td>
<td>• The SensingTask cannot access sensor data directly, but through the UserPrivacyManager (principle of compartmentalization)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use of UserPrivacyManager to give options to users about which data to share with SensingTask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PrivacyPolicyManager component share data with the SensingTask based on user’s contextual rules</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use of the TaskPrivacyManager in the SensingTask module to implement data privacy algorithms (e.g., k-anonymity, l-diversity)</td>
</tr>
<tr>
<td>Data submission</td>
<td>Re-identification attacks</td>
<td>• Use of Orbot client to hide network identifiers when a SensingTask upload data to the server</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SecurityManager in both the MetroTrackTaskAgent and in the SensingTask to encrypt data between mobile client and server</td>
</tr>
<tr>
<td></td>
<td>Contextual attacks</td>
<td>• Use of TaskPrivacyManager in the SensingTask module to implement data privacy algorithms (e.g., k-anonymity, l-diversity)</td>
</tr>
<tr>
<td>Data analysis and sharing</td>
<td>Re-identification attacks</td>
<td>• Data are stored by using unique identifiers in each installation of a SensingTask to store and analyze data. No personal identifiable information is collected by the system from its users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The use of Tor does not allow the system to obtain network metadata (e.g., IP addresses) about the users submitting data</td>
</tr>
<tr>
<td></td>
<td>Contextual attacks</td>
<td>• Use of the ExternalPrivacyManager in the MetroTrack server module to implement data privacy algorithms (e.g., k-anonymity, l-diversity, differential privacy)</td>
</tr>
</tbody>
</table>
The prototype system called MetroTrack. Finally, we evaluated MetroTrack and discussed future research issues that require further attention for the prototype system.

7. ACKNOWLEDGMENTS
We thank the anonymous reviewers for their comments which helped us to improve the content, organization, and presentation of this work. Alfredo J. Perez was supported by the US National Science Foundation and the US Department of Defense’s ASSURE program under award 1560214. Sherali Zeadally’s work was supported by a University Research Professorship Award from the University of Kentucky in 2016.

8. REFERENCES
[23] Li, N., Li, T., Venkatasubramanian, S.,2007. l-closeness: Privacy beyond k-anonymity and l-diversity. In Data Engineer-


ABOUT THE AUTHORS:

Alfredo J. Perez is an Assistant Professor in the TSYS School of Computer Science at Columbus State University (Columbus, GA, USA). He received his doctoral degree in Computer Science and Engineering from the University of South Florida (Tampa, FL) and his bachelor's degree in Systems Engineering from Universidad del Norte (Barranquilla, Colombia). His research interests include privacy, Internet of Things, mobile/ubiquitous computing and sensing, and computer networks. He is a member of the U.S. National Academy of Inventors.

Sherali Zeadally is an Associate Professor in the College of Communication and Information at the University of Kentucky (Lexington, KY, USA). He received his doctoral degree in Computer Science from the University of Buckingham, England and his bachelor's degree in Computer Science from the University of Cambridge, England. His research interests include cybersecurity, privacy, Internet of Things, and computer networks. He is a Fellow of the British Computer Society and a Fellow of the Institution of Engineering Technology, England.
A Pipeline-Based Heterogeneous Framework for Efficient Synthetic Light Field Rendering

Chih-Chen Kao  
National Taiwan University  
d03944003@ntu.edu.tw

Liang-Chi Tseng  
National Taiwan University  
r05922035@ntu.edu.tw

Wei-Chung Hsu  
National Taiwan University  
hsuwc@csie.ntu.edu.tw

ABSTRACT
The research interest of real-time global illumination has increased due to the growing demand of graphics applications such as virtual reality. Recently, the design that combines Image-Based Rendering (IBR) and Ray-Tracing to create Synthetic Light Field (SLF) has been widely adopted to provide delicate visual experience for multiple viewpoints at an acceptable frame rate. However, despite its parallel characteristic, constructing a SLF is still inefficient on modern Graphics Processing Unit (GPU) due to the irregularities. For instance, the issues caused by branch divergence, early-termination and irregular memory access prolong the execution time that cannot be simply resolved by workload merging. In this paper, we proposed a Runtime framework that reorganizes the execution into a pipeline-based pattern with grouping of primary rays. The workloads are later distributed to all heterogeneous cores to increase the efficiency of the execution. With this approach, the number of valid rays can be maintained at a high level with less divergence of paths. Based on the experiment on a heterogeneous system, the maximum throughput for a single GPU becomes 3.12 times higher than the original on average and becomes even higher on systems with multiple heterogeneous cores.

CCS Concepts
• Computing methodologies → Ray tracing; Image-based rendering; Graphics processors; • Computer systems organization → Heterogeneous (hybrid) systems;

Keywords
Ray-Tracing; Light Field; Image-Based Rendering; Heterogeneous System; Global Illumination

1. INTRODUCTION
With the growing demand of Virtual Reality (VR) and computer aided design applications (CAD), the research of real-time global illumination rendering has attracted great attention. However, conventional approaches for photo-realistic rendering, such as Ray-Tracing, have difficulties in providing images at an acceptable frame rate for multiple viewpoints due to their requirement of high computing power. For instances, rendering a view using Ray-Tracing requires tracking the interaction of ray batches with a scene that may consist of millions of geometric primitives with various materials. Rays will bounce several times iteratively in a scene before the process is completed and can generate new rays. Therefore, different from the previous rendering systems that leverage the methods from computer graphics, there are many more modern frameworks that adopt image-based rendering from the field of computer vision.

Image-Based Rendering (IBR) systems generate views of an environment from an array of pre-acquired imagery called Light Field or Lumigraph. The source of the pre-acquired images can be obtained from real cameras or virtually constructed. In computer graphics, Light Fields are typically produced either by rendering a 3D model or by photographing a real scene. In either case, views must be obtained for a large collection of viewpoints in order to construct a Light Field. By extracting appropriate 2D slices from the 4D light field of a scene, new views of the scene can be synthesized. The display algorithms for IBR typically resample and extract an image slice from the Light Field. Thus, they require modest computational resource and are suitable for real-time implementation on end-user devices such as smartphones or VR headsets with restricted power budget. A practical example is demonstrated in Figure 1. In this example, the entire Light Field is constructed in advance and put into the mobile device enclosed in the cardboard cage for demonstrating the VR application. In order to render the display for any particular viewpoint, only the interpolation algorithm is required to be implemented in the mobile device. Another use case scenario as depicted in Figure 2 has demonstrated the capability of interactive rendering by adjusting the focus point dynamically based on the user input. In fact, many algorithms have been proposed to interactively render photographic effects such as variable focus and depth-of-field within a Light Field [10]. Consequently, these demonstrations emphasize the importance of Light Field in modern graphics applications. The problem is, how can we acquire the Light Field in a more efficient way?

A Light Field built from a virtual environment is called Synthetic Light Field (SLF). With this design, the Light Field technology can be further extended to support applications such as virtual game environment or interior design. For instance, NVIDIA Iray is widely used to render physically-based pictures for product advertisement [7, 16]. To achieve
global illumination, the two techniques, Ray-Tracing and IBR, can be combined to produce delicate effects. For example, the sub-images in the Light Field can be rendered by Ray-Tracing algorithms. To maintain the benefits of IBR, the construction of the Light Field and the new views can be separated. Since the sub-images are all pre-computed, the processing can be conducted in one place and sent to another after compression. However, no matter what the application is, constructing a SLF is always intricate. The combination of Ray-Tracing and Light Field will complicate the producing process, increasing its cost with an order of magnitude since there will be N images with billions of rays waiting to be traced depending on the size of the Light Field. Since Ray-Tracing is known as embarrassingly parallel problem, it is often executed on a Graphics Processing Unit (GPU). In this configuration, each work-item in a wavefront of a GPU is responsible to trace a path of a single ray, conducting intersection test with primitives and render the image by accumulating the radiance. To reduce the time complexity to logarithmic scale, a tree-based accelerator is commonly used to accelerate the intersection test. Therefore, the procedure of the test is often referred as Traverse and Intersection (T&I). However, Ray-Tracing suffers from performance penalty while being processed by a GPU due to its characteristic of irregularity [8, 34, 3]. For example, the control flow divergence in a kernel will postpone the execution of the work-items on the non-taken branch during tree traversal due to the nature of SIMD processing. Moreover, in Ray-Tracing algorithm, a specific type of the divergence problem called early-termination also occurred frequently. That is, during the T&I test, a ray misses all the primitive and returns the control back to the host before others have completed the whole process. This results in underutilized hardware resources [27, 25].

To leverage the computing power of GPUs and accelerate the process of building a SLF, one may choose to apply a straightforward approach that merges all the sub-images as one set of data and submit the expanded workload to the GPU simultaneously instead of processing them sequentially. Nevertheless, this method may not achieve the expected performance gain due to the following drawbacks: First, the total amount of workload decreases rather quickly due to early-termination of rays, which reduces the effectiveness of GPU processing. Second, the derived rays from the bounce iterations will break the coherency, which causes more branch divergence to happen especially when they are grouped together. Finally, since derived rays may traverse different paths of the accelerator, it leads to severe memory access irregularity that cannot be hidden by interleaving the wavefront execution.

By analyzing the property of viewing, we found that the computing model can be improved by taking advantages of the characteristics of SLF. For instance, since the projection screens of the camera (viewport) can be similar if the cameras are placed within a small distance, the primary rays generated from different viewpoints can be highly coherent and will traverse similar paths in the accelerator. With this feature, we can exploit the result of T&I from one viewpoint to group another set of primary rays to reduce irregularity. To fully utilize the intersection result, the execution pattern needs to be restructured since it will depend on the order of execution. The method that merges all the primary rays from all the viewpoints cannot be applied. Although falling back the execution flow to a sequential pattern may maintain the order, it will lose the advantage of using GPU and reduce the parallelism. For this reason, we proposed the pipeline-based approach. With this method, the computation of another sub-image will start to generate primary rays immediately and will be merged with the previous one which is entering its second bounce of the iteration. The benefit of this methodology is that it preserves the execution order of primary rays from different viewpoints and could also maintain the number of actively rays at a high level.

In this paper, we analyze the metrics of GPU performance and the design trade-off. The evaluation includes the impact of control flow divergence, and the hit rate of cache. We construct a Runtime system with the proposed methodology that is capable of rendering the sub-images of a Light Field in a pipeline-based execution order. The result leads to performance improvement due to the increase of GPU utilization. We later extend the framework design to support the execution on multiple heterogeneous cores to gain further speedup improvement. The paper makes the following contributions:

1. We proposed a Runtime design for software that reorganizes the execution to become a pipeline-based model. It maintains the number of active rays in the workloads to achieve high GPU utilization with potentially less branch divergence.

2. With careful analysis of our proposed model on a heterogeneous system, we have indicated design considerations of constructing the Runtime for IBR systems that can achieve high throughput by using the same computing resources.

3. We compare the performance improvement on single core GPU and later extended the framework to support multiple heterogeneous cores which further leverages the overall throughput. It proves the framework is scalable for modern heterogeneous architectures.

2. RELATED WORK

The Light Field is a plenoptic function that describes the distribution of radiance or the amount of light flowing in every direction through every point in space [1, 20]. The concept of Light Field Rendering has been proposed in the previous studies [18]. The authors denoted that a display system which requires multiple viewpoints ability can benefit from such rendering method. With the support of the Light Field, a projected image can be reconstructed from any viewpoint and direction. The concept of using the Light Field for rendering is similar to many other approaches such as photon mapping and ray bundle in stochastic global illumination [29, 17, 19].

However, it is also reported that constructing a high quality SLF is very time consuming due to long processing time of
ray tracers. Some studies attempted to address this problem. For instance, methods such as approximation and relative sampling were applied to accelerate the construction of SLF [28]. Our work aims to solve this problem with a different angle which optimizes the rendering efficiency of SLF on Ray-Tracer level.

Several Ray-Tracing-Based rendering systems were implemented on GPUs aiming to gain higher throughput. However, GPUs are vulnerable from branch divergence and Ray-Tracing is irregular by nature. This limits a rendering system to fully exploit hardware potential [5]. To tackle this problem, researchers focus on the T&I stage. In this stage, exploiting the property of coherent ray can be beneficial for performance since those rays have similar behavior when traversing accelerators of a scene. Packet ray tracing, ray sorting and reordering are techniques that aligned with this concept [4, 6, 8, 9, 30]. While these methods focus on single-image Ray-Tracing, we extend these concepts with pipeline-based execution to support efficient SLF construction.

Another branch of studies focuses on the issue of early termination. It happens when some rays running in a GPU wavefront finish their tasks earlier than others. When occurred, the wavefront utilization would become lower, which leads to performance degradation. Novak and Wald discussed the issue and presented methods that regenerate or compact active threads to tackle it [22, 33]. However, the above methods are mainly implemented in a Persistent Thread model that is not widely supported in the current frameworks. The kernels may require complex modification to fit the model.

Different from their approaches, we choose a pipeline-based method to avoid this limitation.

We designed a Runtime that significantly improves the efficiency of building a SLF by designing a pipeline-based method with primary ray grouping. Our system is able to self-calibrate as we gather more information from a scene. Some previous work also tried to design better Ray-Tracing Runtime systems, but their designs are limited to single image. Moreover, without ray grouping, their system cannot fully utilize the hardware potential due to the performance penalties from irregularities of rays [12, 14, 15].

Some researchers have tried to explore the similarity of rays in order to form coherent and regular workloads that are more suitable to be executed by GPUs. For example, Kao et al. have proposed a mechanism that could combine rays based on their characteristics by utilizing an Attribute Table [13]. Moreover, some work focused on finding rays with the same traversal path in order to reduce potential branch divergence that would severely decrease the performance of GPUs [31]. Likewise, we borrowed similar ideas to design the Runtime framework. However, since the number of workloads for constructing a SLF could be many times greater than rendering a single screen in typical Ray-Tracing, we must tweak the design accordingly to fit the execution model in order to minimize the overhead being introduced.

Previous studies have introduced the importance of utilizing every heterogeneous core in a system and have designed different types of methodologies and algorithms for task dispatching. For instance, Pajot presented a hybrid bidirectional path tracing implementation which utilizes techniques such as double buffering, batch processing, and asynchronous execution to balance tasks between CPU and GPU [23, 24]. Tzeng at el. experienced several task management techniques, task-donation, and task stealing methodologies for irregular workloads [32, 11]. Furthermore, Kao et al. have also introduced the concept of Heterogeneous Queue which is derived from the Thread Pool model in order to achieve fine-grained load balancing [15]. However, these methods often require the support from special hardware configurations such as Heterogeneous System Architecture in order to achieve the full potential. In contrast, we adopt a simpler but still effective approach that utilizes a feedback-guided profiling mechanism to offload some workloads to CPUs.

3. ALGORITHM DESIGN

3.1 The Design Philosophy

The SLF in this implementation is created from a large array of rendered images projected from a grid of viewpoints (cameras). The sub-images of a SLF is rendered by using a Path-Tracing algorithm that generates rays, performs T&I operations with the objects in the scene to find the closest intersection point, and integrates the radiance to color the pixels. The implementation includes components such as integrators, samplers, materials and BRDFs. The design is broadly derived from that of the PBRT [26]. In the Path-Tracing algorithm, a ray is traced several times before termination. We define a variable called bounce to identify how many times a ray is traced inside a scene. Since each
stage of the bounce is processed inside an individual loop iteration, we further define the current iteration that processes the given bounce as \textit{bounce iteration}.

While rendering the Light Field, the workloads of sub-images are mapped to the N Dimensional Range (ND-Range) in OpenCL kernel which is usually called global size. The fundamental design philosophy of the pipeline-based execution is based on the observation that the execution time of the merged workloads will grow sub-linearly according to the size of ND-Range compared to the sequential counterpart. Figure 3 demonstrates this behavior. In this setting, we compared the accumulated execution time of rendering the same number of sub-images in a Light Field. The setting \textbf{Sequential} renders the images one by one in serial whereas the \textbf{Merged} configuration compacts all the data into one kernel launch. This phenomenon denotes that it is beneficial to combine more workloads since the throughput will increase due to the rise of parallelism. The reason of such improvement is that it enhances the effectiveness of internal wavefront (warp) scheduling. Note that the GPU can hide overheads, such as memory access latency, due to the property of vertical multithreading if there are sufficient amount of interleaving wavefronts.

However, the speedup improvement is limited since the derived rays will also increase the occurrence of control flow divergence. Hence, the execution pattern needs to be restructured to achieve better performance result. In order to improve the performance while keeping the benefit of merging the workloads, the Runtime must minimize the irregularities that would happen in each bounce iteration. This can be achieved by primary ray grouping.

3.2 The Pipeline-Based Execution Model

Due to the early termination of rays, the number of active rays will decrease sharply after each bounce iteration especially in open scenes. This phenomenon reduces the effectiveness of GPU computing due to the decreasing level of parallelism. Merging all the workloads of the entire views together cannot solve the problem since they will still follow the same decreasing trend. With this setting, the total number of rays will be insufficient to form a proper ND-Range eventually. In addition, the size of the workload of each bounce iteration will differ drastically. For example, based on the observation, we found that in the usual case the size of the primary ray will exceed the sum of the workload size of the rest of the bounce iterations.

To resolve this issue, the pipeline-based execution pattern is proposed. In this algorithm design, primary rays for a set of sub-images will be generated before each bounce iteration. Figure 4 illustrates the execution pattern of the pipeline-based algorithm. It this figure, four snapshots of the same ray buffer with time line ordering is plotted. The letters beside each ray buffer indicate the current state. The bounce iteration of each set of the primary rays is numbered at the bottom right corner in the segments.

We use an example to demonstrate the mechanism as depicted in Figure 4. Assume that the bounce iteration is configured to 3. With this setting, the ray buffer will be segmented into 3 parts. To construct a Light Field with 12 sub-images, we may choose to merge every 3 of them in to a set, and process the sets in a pipeline fashion. In this case, the 12 sub-images are partitioned into four sets which are colored as blue, orange, yellow and green. The primary rays of 3 images in the blue set will be placed at the first segment of the ray buffer in state $A$. In state $B$, the primary rays of another 3 images in the orange set will be placed at the second segment after referencing a lookup table. The lookup table is used to group similar rays for reducing irregularities in the kernel. The mechanism of table referencing will be elaborated in the next subsection. In state $C$, the output of three images from the blue set will be generated and transferred to the Light Field structure since the bounce iteration for this set is completed. After the termination of state $C$, the data from the blue set are no longer required. Therefore, the new primary rays of another 3 images in the green set will overwrite the data of the blue set in state $D$. Consequently, the pipeline design interleaves the execution of bounces from different sub-images. Note that the placement of the primary rays from distinct views adopts a circular pattern. Therefore, the memory footprint is restricted in the buffer area that is allocated at the beginning. No extra memory allocation is needed during the pipeline execution. The rest of the execution will follow the same pattern. In addition, an image usually requires multiple samplings before completion. We can start another process of sampling for the previous sub-images (e.g., the blue set) and interleave it with the ongoing pipeline to further utilize the buffer space.

The size of the required ray buffer depends on the number of bounce iterations and the number of views that are concurrently being processed. We first extend the size of the ray buffers to store rays from different viewpoints. The original configuration that merges all the workloads requires a buffer with the size of original ND-Range times the number of all sub-images. In the pipeline configuration, the buffer size will become original ND-Range times the number of bounce iterations times the number of sub-images in a set that are concurrently being rendered. For instance, in the above case the ray buffer size becomes $N D R a n g e \times 3 \times 3$. The generated primary rays will therefore be combined with the derived rays from the previous bounce iterations. Since the number of added primary rays is greater than the de-
rived rays, it helps preserving the number of actively valid rays in the kernel execution.

### 3.3 Grouping with Similar Rays

Increasing the number of active rays may cause branch divergence that leads to performance penalty if they are simply combined together. Fortunately, by analyzing the pattern of generated rays, we found that the rays generated from individual stages possess a high degree of similarity since the camera position of views are closely related. Therefore, we can group rays from the previous result of T&I test. Thus, a table-based data structure is constructed in the Runtime system to fulfill this purpose. It correlates the orientations of the rays with the ID of the intersected primitives.

In the pipeline-based execution pattern, whenever a set of rays from another view is generated, they will be sorted by comparing the direction and origin according to the records in the table. After that the grouped result will be placed to the corresponding segment in the ray buffer. The table structure located between state A and state B in Figure 4 depicts this mechanism. Figure 5 further demonstrates the process of sorting and grouping in detail. After sorting, the rays with similar intersection points will be grouped together by the primitive IDs.

The algorithm of ray grouping is demonstrated in Algorithm 1. In the actual implementation, each ray will be converted to weighted sum that is parameterized by its origin and direction in order to reduce the complexity of the sorting process. Of course, there will have rays that cannot be grouped since they are not in the range of the previous view. These rays will be placed at the bottom of the ray buffer after the sorting process. However, their intersection result will still be recorded in the table for future references. By doing so, the records of all the rays generated from all viewpoints will eventually be stored in the lookup table.

To avoid data transfer overhead during sorting, a doubly-indexed approach is applied. Figure 6 demonstrates the layout of a doubly-indexed array. Inside the structure, we maintain two sub-arrays: the ray buffer array that stores all the meta-data of rays and the index array which records the location of rays with unique integers. The ray buffer and the index buffer can be allocated on the region of Shared Virtual Memory (SVM) that is supported in both OpenCL 2.0 and HSA [21]. With this setting, the two arrays are accessible directly by both the CPU and GPU. Thus, no extra buffers are required to be allocated for data movement during the grouping process. The grouping algorithm only sorts the index array so that the meta-data do not need to be swapped. The correlation of the index array and the ray buffer is shown in Figure 4 between state B and state C.

Due to the fact that the majority of the rays are all casted from the camera (eye-ray) in the pipeline execution, if these rays can be successfully grouped, then we can guarantee a large portion of the workloads will follow the expected traversal path. This behavior improves the performance significantly since the potential branch divergence can be greatly reduced. Moreover, by grouping the rays with similar traversal paths, we can decrease the size of memory transfer since node data in the accelerator structure will be shared by many work-items.

The main part of the grouping operation is basically a table lookup and indexing process as shown in Figure 5. The execution of such operation for each ray will be independent to each other. Thus, due to its embarrassingly parallel characteristic, the lookup process can be implemented as a GPU kernel. This approach greatly reduces the runtime overhead.

### 3.4 Offloading Workloads to CPU Cores

We further extend the proposed Runtime design to support multiple heterogeneous cores. In order to further increase the throughput, we must utilize all the available system resources and processors in the system. This design consideration inspires us to implement a workload distribution mechanism that dispatches the computational tasks to different cores based on their characteristics and related throughput. The simplest approach is to utilize the flexibility of the OpenCL framework. Due to the fact that an OpenCL kernel can be compiled into binaries of different types of heterogeneous configurations, we can preallocate dedicated threads to handle the execution flow for each core. Each thread should initialize the corresponding core by triggering...
the OpenCL API with distinct configurations accordingly. In other words, there will be a one-to-one mapping of each thread and core in the system. After that, each thread is responsible for the execution status of the corresponding core. Therefore, with this setting, the next question becomes how to decide how many workloads should be assigned to individual cores.

The workload dispatching algorithm is very crucial to the performance in this setting. With a suboptimal assignment, the thread that finished its execution early will have to wait for other threads that have not yet finished the tasks being assigned to them, which decreases the overall performance. Therefore, we must carefully dispatch the workloads based on the relative throughput of the cores so that all the threads could finish their tasks at roughly the same time. This goal is accomplished by the following designs: prioritizing and profiling. The fist approach is to utilize the grouping results. We divide the task from the pipeline stages into multiple subtasks based on the assigned groups and prioritize the subtasks with more number of rays to GPUs. The rest of the subtasks are reserved to be executed by CPUs since they are more likely to generate internal branch divergence. The second approach is to calibrate the number of subtasks taken by each thread at runtime. Since the number of subtasks executed by each core can vary during runtime based on the situation, we implement a feedback mechanism with a profiler to monitor the throughput of each core dynamically. The profiler records the related throughput of individual cores and calculates the optimal ratio for workload distribution. The profiling data will later be used to adjust the execution pattern regarding the assignment of subtasks before the next run. Figure 8 illustrates the concept of workload adjustment implemented in our framework.
4. EXPERIMENTS

The experiment was conducted on an AMD Kaveri A10-7850K with 8GB of RAM. The proposed Runtime is based on RadeonRays with AMD APP SDK 3.0 for OpenCL [2]. The number of bounce is set to 5. The resolution of sub-images of a Light Field is 800x600. Five scenes are used to generated the Light Field by using three types of configuration (Sequential, Merged and Pipeline). The test scenes are illustrated in Figure 9 as examples. The setting Sequential represents the method where all the sub-images of a Light Field are rendered sequentially. The Merged indicates the method that merges all the workload of sub-images and compute them simultaneously. Finally, the Pipeline represents the proposed method. The path trace integrator is unidirectional (rays cast from the virtual viewpoint) and uses Monte Carlo sampling, Russian-roulette termination, and local evaluation of direct illumination.

4.1 Impact of GPU Utilization

We compare the improvement of GPU utilization by measuring the number of active rays that are simultaneously executed by a kernel. Figure 10 demonstrates the result. In an open scene such as BMW, the number of rays decreases sharply due to the effect of early-termination. With the support of pipeline-based execution, the number of active rays can be maintained at a static level since the rays for another view will be created in the ray buffer and will join the computation with the previous derived rays. Thus, the size of ND-Range can be large enough for a GPU to increase the effectiveness of internal scheduling since the active rays can be filled into more wavefronts. As a result, the hardware utilization can be improved.

4.2 Impact of Branch Divergence

We measure the impact of control flow divergence by comparing the performance of the TkI test. Figure 11 illustrates the result. As shown in the figure, the performance speedup of the Merged configuration is insignificant compared to the Sequential setting (only 6.2% in average). This is caused by control flow divergence: Although merging the workload creates more active rays, the internal branch divergence will delay the execution time. The statement is proven by the fact that both Sequential and Merged setting possess a very low VALU (Vector Arithmetic Logic Unit) percentage. Notice that a low VALU percentage indicates that some of the work-items are frequently postponed due to the irregular control path. In contrast, the Pipeline configuration can achieve a relatively high VALU percentage since most of the primary rays are grouped. Therefore, the traversal path of rays in a group are mostly identical.

4.3 Impact of Memory Access

We demonstrate the benefit of ray grouping by showing the total memory fetch size. As shown in Figure 12, the memory fetch size is reduced since there are more work-items in a wavefront that shared the same path after ray grouping. In this case, the node data for the accelerator need to be fetch only once. This result also leads to a high cache hit rate.
With the support of memory-level parallelism in GPUs, this phenomenon can further improve the efficiency of execution since, after the reduction on memory transfers, the latency of memory access can be hidden by the mechanism of vertical multithreading in a GPU.

4.4 Grouping Efficiency

We measure the number of grouped primary rays to identify the effectiveness of grouping. Figure 14 illustrates the result. The result shows that the number of grouped primary rays is much greater than the non-grouped derived rays. Almost half of the rays are primary rays. Thus, grouping the primary rays into regular patterns can lead to significant performance boost. Another experiment shows the grouping efficiency while altering the viewpoints. In this experiment, five cases of Light Fields were created by altering the camera positions (viewpoints) of images with distinct angles. Case 0 has no angle difference. We add 10 degrees starting from case 1 to 4. The result in Figure 13 indicates a higher hit rate of grouping will lead to better performance. Notice that the sub-images need to be sampled multiple times to fulfill the quality requirements. The figure represents the result in stable state after 256 samples per pixel.

4.5 Workload Distribution

Figure 15 illustrates the percentages of workload distribution of each heterogeneous core. The data are recorded after several iterations of running until the partition is stabilized. In open scenes such as Dragon, Mixed and BMW, since the ray paths are mainly identical in a wide range of the area mapped on the screen, the number of internal branches is relatively lower. Therefore, these types of workloads can be efficiently executed by GPUs with higher throughput. In the figure it shows that the GPU actually consumes more workloads, which reflects this phenomenon. On the contrary, the numbers of workloads of CPU and GPU in a complex scene such as Sponza and Hair are almost equal.

4.6 Performance Speedup Comparison
4.6.1 On a Single GPU

Figure 16 illustrates the measurement of the overall performance speedup. The comparison is based on Sequential. According to the figure, the speedup of Merged is very limited in many scenes. It gives almost no improvement in scenes such as Dragon (12.81%), Fairy (2.48%) and BMW (7.17%). In contrast, Pipeline shows its effectiveness in all cases. It increases the throughput by 148% in average whereas the Merge approach can only achieve 37%.

The Merged configuration in the Mixed scene can achieve 127% of the speedup since it contains a wide areas of blank that are continuously aligned. The rays being emitted towards these areas will terminate immediately. Thus, the Merged approach in this setup can successfully achieve large speedup by aligning the blank areas. However, the Pipeline approach can boost the performance even further.

4.6.2 On Heterogeneous Cores

Figure 17 illustrates the measurement of the overall performance speedup on multiple heterogeneous cores. In these test cases, the pipeline algorithm is used. According to the figure, the overall speedup on heterogeneous cores became higher than that on the single GPU. The peak record in the diagram reaches 338.12% in the case of the Sponza scene. Furthermore, the performance improvement in scenes such as Sponza and Hair is greater than that of other scenes. This is because these scenes are more complex and the relative throughput of the CPU is very close to the GPU. Thus, more workloads are offloaded to the CPU during computation. In other cases, almost 80% of the workloads are execute by the GPU. The recorded data have shown that the proposed framework is highly extendable and could effectively support heterogeneous systems with multiple cores.

5. CONCLUSION

In this paper, we proposed a pipeline-based Runtime framework that is capable of improving the utilization of GPU and reduce pitfalls that caused performance degradations. It can also offload workloads to other cores such as CPUs based on the relative throughput of them in order to increase the overall execution speed. To improve the performance of GPU, the Runtime generates primary rays for another view and merges them with the derived rays from the previous bounces of different views to maintain the number of active rays at a stable level. Before the primary rays are introduced, they are classified with the previous result from T&I due to their similarity to reduce potential irregularities. With this pipeline-based approach, the occurrence of control flow divergence and memory fetch size can be greatly reduced. The effect results in an overall performance boost especially for open scenes. According to a series of measurements, the maximum throughput becomes 3.12 times higher than the original on a single GPU. The Runtime framework can be further extended to support multiple heterogeneous cores. With this setting, the performance is further improved and the maximum throughput becomes 4.38 times higher than the original.

6. REFERENCES


ABOUT THE AUTHORS:

Chih-Chen Kao received his Ph.D. degree in Computer Science from the National Taiwan University, Taipei, Taiwan, in 2017. Currently, he is a Graphics Runtime Intern at Intel with a demonstrated history of working mostly as a freelancer in the computer software industry. From 2011 to 2012, he was invited to participate in the OPTIMI (Online Predictive Tools for Intervention in Mental Illness) program, which is a research project funded by the European Union's 7th Framework Programme Personal Health Systems, at ETH Zurich, Switzerland. His research interests include Computer Graphics, Physically-Based Rendering, Heterogeneous Computing and Parallel Computing.

Liang-Chi Tseng received his B.S. degree in computer science from the National Chiao-Tung University, Hsinchu, Taiwan, in 2016. Currently, he is a master student at the Department of Computer Science and Information Engineering at National Taiwan University, Taipei, Taiwan. In 2016, he was a research assistant about virtual reality and parallel computing in Institute of Information Science, Academia Sinica, Taiwan. His research interests include computer system and parallel computing, and computer graphics.

Wei-Chung Hsu received his Ph.D. degree in computer science from the University of Wisconsin, Madison, in 1987. Currently, he is a professor in the department of computer science and information engineering, at National Taiwan University, Taiwan. From 2010 to 2013, he was a director of the Institute of Computer Science and Engineering at National Chiao Tung University. From 1999 to 2009, he was a professor in the Computer Science and Engineering Department at the University of Minnesota, Twin Cities, USA. From 1993 to 1999, he was a Runtime optimization architect in the California Language Lab at Hewlett-Packard, CA, USA. Prior to joining HP, he was a senior architect at Cray Research, in Chippewa Falls, Wisconsin. His current research interests include virtualization, dynamic binary translation and optimization techniques.
Applying Software-Defined Networking to Minimize the End-to-End Delay of Network Services

Tommy Chin  
Rochester Institute of Tech.  
Rochester, NY 14623 USA  
tommy.chin@ieee.org

Mohamed Rahouti  
University of South Florida  
Tampa, FL 33620 USA  
mrahouti@mail.usf.edu

Kaiqi Xiong  
University of South Florida  
Tampa, FL 33620 USA  
xiongk@usf.edu

ABSTRACT
There is a grand challenge to transfer the gigantic volume of data generated from a variety of smart cities applications. It is an urgent need to efficiently and effectively transmit such time-sensitive data in a wide-area communication. In this paper, we propose an approach by leveraging Software-Defined Networking (SDN) to develop a communication solution for data transfers in smart city applications. Our approach is Quality of Service (QoS)-aware that minimizes the end-to-end delay for data transmission in an SDN infrastructure by using a Timestamp Recording method to compare the arrival and departure of flows and packets over a period. Finally, we evaluate our solution on the Global Environment for Network Innovations (GENI), a real-world federated testbed. We first compare the Timestamp Recording to other common delay measurement techniques. Our analysis demonstrates its effectiveness at scale. Additionally, we carefully examine the accuracy and efficiency in minimizing end-to-end delay delivered by our SDN solution with contrast to a non-optimized SDN environment. Our experiments demonstrate the proposed solution effective and efficient in delivering data with an optimal delay. Particularly, our solution reduces about 22% in end-to-end-delay compared to a non-optimized SDN environment.

CCS Concepts
•Networks → Network experimentation; Network performance analysis; Network measurement;

Keywords
End-to-End Delay; Path Selection; Priority Traffic; Software-Defined Networking (SDN);

1. INTRODUCTION
Building an efficient communication system is not only important but also necessary in a variety of application services in smart cities. For instance, a next-generation power grid system, as one of the most critical applications in smart cities, requires an efficient networking infrastructure to support the efficacious generation, transmission, and distribution of Phasor Measurement Unit (PMU) data. Furthermore, in such a communication system, the end-to-end delay can affect the performance of entire grid systems, causing power losses, and possibly equipment damage [1, 23].

The end-to-end delay plays a critical role in the delivery or transfer of data between the source and the destination within end-to-end devices. The information exchanged between these devices is often beneficial or valid just for specific time slot windows. Hence, if the delay associated with the communication system surpasses a predefined time window, the information could no longer serve its objective. Thus, the key components to efficient network traffic forwarding in an environment are the resiliency of a network infrastructure to flash crowd traffic (burst), the ability to minimize an end-to-end delay and to provide the highest priority where Quality-of-Service (QoS) is guaranteed [18, 19, 24]. As users are broadly interested in network service processing time instead of network throughput, the end-to-end communication among end-user devices is required to guarantee a maximum QoS level. Thus, accurately predicting the perceived QoS for a user allows service providers to not only assure QoS but also avoid over-provisioning to meet a user’s requirement. Because of a variable burst of network traffic derived from customer requests, the dynamic assurance of a minimal end-to-end communication delay will not be an easy task.

Unlike a traditional network, a Software-Defined Networking (SDN) environment utilizes SDN controllers to provide a centralized and holistic view of the network [7]. Besides, the key features of SDN include, but are not limited to, flexibility and programmability [14, 15, 17]. These features are used to ensure different QoS requirements. Policy Cop [5], VSDN [16], and other QoS-based routing frameworks [10, 21] utilize a centralized routing approach to improving video delivery within end-to-end entities. Herein, the limitation of these applications is that they implement solely those QoS management functions that are already available in the newest versions of some SDN controllers (such as Floodlight and OpenDaylight). In this paper, we leverage the flexibility of an SDN controller to implement a Timestamp-based framework that takes into account the shortest path selection. This framework measures and optimizes the end-to-end delay for a real-time communication system in SDN environments, where such a QoS metric is of great importance to realize end-to-end applications, for instance, e-learning [12]. Although the proper development of such end-to-end applications is required to control the communication delay.
between end-to-end devices [25], there are still two significant challenges to take into account. The first challenge is the simultaneousness, implying that all entities need to be time-synchronized. The second challenge is timeliness where the transmission of data has a precise and sensitive deadline. Furthermore, prioritizing application services is also important. There have been many solutions aimed to minimize end-to-end delay in a traditional network environment such as OpenQoS [9], MCUP [26], and an SDN controller for cloud gaming [2]. However, to the best of our knowledge, none of them has addressed the end-to-end delay for a network that handles different priority traffic at burst intervals for network traffic with different priorities.

Efficiently measuring the end-to-end delay between two end devices is critical in this research. Since ‘Traditional Ping’ is a basic approach to calculating delay within network-enabled entities by deploying Internet Control Message Protocol (ICMP/Ping), we consider this approach as a baseline value due to ICMP’s fundamental nature in the end-to-end delay measurement. The delay between a source and a destination is computed as one-half of the round trip time (RTT) of a message from and returned to the source. We further consider ‘Packet Probing’ as our second approach for the calculation where the end-to-end delay is computed based on the Round-Trip Time (RTT) of Link Layer Discovery Protocol (LLDP) and Broadcast Domain Discovery Protocol (BDDP) accordingly. However, there is a significant fault associated with Packet Probing, which is the excess of network traffic generated by these two network protocols and the period to examine delay. Our third approach is Timestamp Recording where Open vSwitch records the arrival and departure time of a per-packet flow where the delay is calculated base on the sending and receiving time over a set of packets. We extensively examine the effectiveness of Timestamp Recording, Traditional Ping, and Packet Probing on the ExoGENI [4], one of the Global Environment for Network Innovation (GENI) testbeds [6]. Our experimental evaluation demonstrates that Timestamp Recording outperforms the other two measurement approaches regarding measurement accuracy. In the end, we study and examine Dijkstra’s algorithm on an SDN controller for prioritized application services using Timestamp Recording where we aim to select an optimal path in the network. Our ExoGENI experiments show the applicability and effectiveness of Timestamp Recording in the optimal path selection.

The rest of our paper is organized as follows. Section 2 discusses related work. Section 3 gives our research methodology. Section 4 outlines the experimental results of this research. Finally, we conclude our work with future work in Section 5.

2. RELATED WORK

QoS is the non-functional attribute of a service such as an end-to-end communication delay [3]. Satisfying such a QoS metric is of great importance towards customizing an efficient and reliable network in smart city applications. For example, in power grid systems [3], the communication delay between a power grid substation and its controlled electronic device needs to be in milliseconds to maintain the balance of power flow among power lines.

The end-to-end network delay has been widely studied [9, 20, 22]. While Xiong [20] studied the bandwidth allocation to ensure the guarantee of QoS including the end-to-end delay, Xiong, et al. [22] presented an approach to measuring the QoS metric in cloud networks. Likewise, OpenQoS [9] leveraged OpenFlow to deliver multimedia network traffic. Although it can deal with the dynamics of network traffic and provide the shortest path in terms of the QoS metric, the implementation of OpenQoS [9] is limited to a very small scale local area network.

In OpenQoS [9], SDN controllers repeatedly update their data planes to optimize network utilization and provide dynamic QoS routing for data delivery. However, these repetitive updates in OpenFlow networks by SDN controllers maximize network utilization and may cause serious transient congestion and packet loss. In MCUP [26], authors formulated the update of data planes as an optimization problem and solved it heuristically. However, they only evaluated their SDN-based approach through Mininet simulation.

Amiri, et al. [2] employed SDN to adaptively disperse the cloud game traffic load among different routing paths based on end-to-end delays and jitters for cloud players. Under QoSFlow [13], researchers examined the way to minimize end-to-end delay from the perspective of packet schedulers using Linux kernels. However, their approach provided the ability to individually fine-tune each SDN device from a system-level point of view, and to optimize QoS mechanisms. Although QoS provides the ability to manage traffic type regarding priority, OpenTCP [11] fine tunes TCP communication on SDN topology. Under OpenTCP [11], QoS was not addressed in the performance of their design by real-world experiments conducted under their analysis. Instead, we leverage end-to-end delay measurement approaches with SDN and implement them in a real-world at-scale virtual network using GENI computing resources across multiple sites, through the Internet.

This paper is an extension of our previous conference paper [8]. It gives a comprehensive examination of the proposed SDN framework for ensuring the end-to-end delay minimization of network services. Furthermore, we investigate the optimal path selection in our SDN framework and the behavior of the optimized SDN controller under such a scenario with varying types and quantities of network traffic.

3. DELAY MINIMIZATION METHODS AND PATH SELECTION

A computer network supports and facilitates the transportation of data, ranging from medical information to multimedia TV shows. The elevation of urgency or the sense of prioritization of information has much consideration in the design of a network and includes, but not limited to, throughput, latency, and end-to-end delay. This study evaluates three methods of end-to-end delay: Traditional Ping, Packet Probing, and Timestamp Recording with considerations modeled from the previous three key factors. SDN provides two major contributors to the design and implementa-
3.1 Controlling a Network

A controller in an SDN environment projects and establishes the final choice in determining path selection, optimizing a network, and handling prioritized data. Other abilities and tasks an SDN controller executes include, but not limited to, the calculation of end-to-end delay and selecting the optimal path in a complex network environment. The three methods of end-to-end delay measurement show a common trait of packet flow analysis using either ICMP, LLDP, BDDP, or statistical calculation in time stamping. The controller in the SDN environment utilizes either one or a combination of multiple end-to-end delay measurement techniques to efficiently perform the optimal path selection for network communication and traffic direction. One challenge to address in the analysis of each end-to-end delay measurement is a comparison to the baseline value.

The solution to the challenge above is to build a non-optimized controller where the SDN system does not consider any delay measurements, and all links in the network are treated equally without considerations of traffic congestions, bottlenecks, or bandwidth. Floodlight, the deployed open-source SDN controller in our work, in its default configuration adopts the behaviors of Packet Probing. The use of LLDP and BDDP facilitates the discovery and latency measurement of paths and links in a network. The non-optimized controller removes the procedure to evaluate link latency and treats all paths equally while utilizing the discovery processes to determine and explore the network topology shape. Figure 1 shows a sample representation of an SDN infrastructure deploying the three techniques of end-to-end delay measurements evaluated in our paper.

3.1.1 The Traditional State Control: Round-Trip-Time

ICMP or ping is a ubiquitous technique to measure the latency between two given points on a network. The Round-Trip-Time (RTT) is the measurement of latency, typically in the form of milliseconds, where a sender records the necessary time for a message to be sent to a recipient and then returned with a particular ICMP response. Figure 1 demonstrates an example configuration where two nodes ping one another. The latency on the link between the two switching devices is defined as one-half the RTT (i.e., RTT/2). Notably, one flaw in this configuration is the consideration of bi-directional or full-duplex traffic for path selection and that the latency measurement defines both directions for the controller. Additionally, this technique does not consider the latency between end-point devices and the switch, and that halving the RTT value will be considered the end-to-end delay.

Under Figure 2, nodes N1 and N2 are clients in a small network topology, S1 and S2 are SDN switches, and Controller is an SDN Controller managing S1 and S2. Within the figure, N1 transmits an ICMP message destined to N2. This ICMP message is received by N2 after a series of packet forwarding by S1 and S2. The ICMP message is then returned to N1 by N2 through S2 and S1. Node N1 calculates the delay based on one half of the ICMP measurement value. Although RTT using ICMP is a fundamental method of measuring end-to-end delay, Packet Probing provides a more accurate measurement through its use of LLDP and BDDP.

3.1.2 The Common State Control: Packet Probing

Packet Probing is one of the more ubiquitous methods to measure end-to-end delay as one of the essential characteristics is the ability to identify the shape of a network with latency measurements of each network path. Floodlight adopts Packet Probing as its defacto method for link discovery and path selection. In an SDN environment, Floodlight jointly deploys both LLDP and BDDP for identifying networking links. Additionally, LLDP and BDDP use multicast and broadcast respectively to identify neighboring network devices from SDN switches.

In Figure 4, an SDN topology is depicted where nodes S1 and S2 are SDN switches, and Controller is an SDN Controller. In the initial transmission, an LLDP or BDDP packet is encapsulated into an OpenFlow Type 13 message and forwarded to S1 for processing from Controller. Within
Figure 3. A visual representation of the experiments conducted on ExoGENI testbed for the proposed SDN framework. All the nodes in this figure are instantiated across multiple hypervisors located in distant regions. Dashed lines represent controller communication and they are individual links to the SDN controller from their respective SDN switches for OpenFlow messages.

Figure 4. Delay is estimated based on Packet Probing techniques where measurements are independent of network client activity.

$S_1$, the OpenFlow Type 13 message is de-encapsulated, and the LLDP or BDDP is sent out to $S_2$. Depending on the configuration of the SDN environment, the LLDP or BDDP packet maybe dropped upon transmission at the POSTROUTING phase of the SDN switch. Netfilter hooks will show the transmitted packet and logged for statistical analysis, but the receiving device will not see the LLDP or BDDP. Instead, OVS will retransmit the LLDP or BDDP packet as a 0x8942 packet with a payload of the LLDP or BDDP content. The receiving device, $S_2$, will interpret LLDP or BDDP messages and directly forward the information to the Controller, encapsulated in an OpenFlow Type 10 message. Delay is calculated based on the trip time of the entire process from $S_1$ to $S_2$ as represented in equation (1).

$$\text{Delay}(S_1, S_2) = T_{\text{Total}} - T_{S_1}/2 - T_{S_2}/2 \quad (1)$$

Additionally, delay measurements between Controller to $S_1$ and Controller to $S_2$ are subtracted from the trip time in order to obtain the pure latency value of the network link. To clarify our approach, Figure 5 expresses a packet diagram of the Packet Probing approach.

As shown in Figure 5, an Optional Type-Length-Value (TLV) is implemented for Packet Probing where epoch time is inserted into each LLDP packet. This epoch time is calculated from the perspective of the SDN-Controller, and the packet will traverse through two SDN switching devices before returning to the SDN-Controller. Using this method, we can obtain the inter-switch latency for each link. BDDP will have a similar approach for a topology that have mixed switching devices of SDN to non-SDN peripherals.

Within (1), $T_{\text{Total}}$ is obtained through Packet Probing while $T_{S_1}$ and $T_{S_2}$ is measured on the response of OpenFlow messages. Essentially, as the Controller and respective SDN switching device communicates using OpenFlow messages, TCP-based communication is established. Delay is measured based on the initial SYN flagged TCP message and the responding SYN-ACK flagged TCP message. We divide the value by two to obtain an estimated one-way delay measurement on a bi-directional network path for the links $T_{S_1}$ and $T_{S_2}$. Upon subtracting $T_{S_1}$ and $T_{S_2}$ from $T_{\text{Total}}$, we can obtain the inter-switch latency. BDDP is an asset in Floodlight for identifying hybrid networking paths of both SDN and non-SDN switching devices. Following a similar scheme to LLDP, BDDP facilitates the discovery of external links, or links that traverse over multiple physical network environments. The joint configuration of LLDP and BDDP in
3.1.3 The Timestamping Control

To identify our approach of Timestamp Recording examination, Figure 6 shows an example topology that illustrates the process. Herein, the delay between two SDN switches is calculated based on the time difference between the sending and receiving switches. Unlike Packet Probing, Timestamp Recording does not interact with end-user devices and therefore utilizes a trusted system between the Controller and SDN device.

\[
\text{Delay}(S, R) = [t_R - (t_{R2} - t_{SZ})] - t_S
\]

Under equation (2), \(t_R\) and \(t_S\) are the received and sent timestamps at SDN switching devices per network link. Delay is measured where \(n\) is the number of packets on a given network stream, \(T_{R2}\) and \(T_{SZ}\) are timezone values for the sending and receiving switching devices. Moreover, the interval of time to examine latency of each link is pre-established administratively. Specifically, to update the metric of delay per link within the SDN environment, an interval value in seconds will need to be set to measure the delay.

To implement such design, we will examine modifying key components to our Floodlight-based SDN Controller with regards to per link latency.

3.2 Path Selection: Dijkstra’s Algorithm

Path selection is a key component to end-to-end delay. To select the optimal path for our SDN-based services, we utilize Dijkstra’s algorithm. Under the approach, a network topology has a finite number of combinations constructed from Dijkstra’s algorithm. The calculation time to compute the number of combinations can be strenuous to any device with regards to system resources and performance. To improve the feasibility of minimizing end-to-end delay, we use Dijkstra’s algorithm on a tuple approach. Under this approach, only the top three paths will be selected to determine the optimal path and therefore reducing the amount of system resources needed to calculate an entire network at scale. To construct the tuple for Dijkstra, we will initially sort the set of all link combinations to the sum of lowest latency value. For each new flow injected into the SDN topology, Dijkstra will be recalculated based upon the end-to-end delay measurement technique. Quality of Service (QoS) plays a significant role in minimizing delay. When a particular traffic flow is identified as a highest priority, the traffic flow will be processed with the lowest latency.

3.3 Prioritizing Network Services

Inter-domain and local network communication present a challenge when traffic of many natures, whether high (time sensitive) or low priority, are mixed in a network. Enabling QoS on suitable SDN devices provides an examination of data communication origins with identification processes to determine whether a particular traffic set should be considered time sensitive. The support of QoS has limitations based on the version of OVS, and specifically the OpenFlow protocol. The construction of a topology to evaluate the three methods of end-to-end delay measurement this study
presents has considerations to factor time-sensitive communication.

4. EVALUATION

Figure 7. Wide-area communication at scale does not limit itself to a single geographical region. GENI testbed allows inter-network domains for inter-region communication to aid in conducting real-world experiments over public Internet traffic.

Conducting a comprehensive experimental evaluation requires the construction of a topology with three properties. The outline of the properties includes, but not limited to, (1) an environment with SDN switching and routing devices, (2) a software module that emulates network traffic, and (3) a network with multiple links and paths with changeable throughput and loss. This study employs Global Environment for Network Innovation (GENI) [6], a real-world federated, heterogeneous testbed solution. ExoGENI, a component within GENI, provides the ability to construct a topology with the previously listed factors with an additional benefit to allow inter-domain network communication over the Internet to address areas of realism.

To evaluate the end-to-end delay among multiple sites, we selected ExoGENI nodes from Lexington, Kentucky, Clemson, South Carolina and Salt Lake City, Utah as shown in Figure 7 and built using a stitched topology. The stitching feature in ExoGENI testbed allows connecting resources provided by multiple aggregates, such as cities, into a coherent network. The architectural design of the topology uses Open vSwitch (OVS) that supports OpenFlow 1.3. We further select Floodlight 1.2 as our SDN controller since it supports both physical and virtual OpenFlow switches, has the capabilities to handle both OpenFlow and non-OpenFlow networks and provides QoS features.

4.1 The Uncontrolled Network State

In our initial experiments, we conducted numerous tests without any optimization or end-to-end delay algorithm implemented to examine the resiliency of the network, during scenarios of a massive burst of network traffic. We utilized the designed non-optimized controller that was previously described in Section 3.

For the baseline study of the non-optimized configuration, Figure 10 demonstrates the intercommunication of two nodes where one is a client, and the other is a server. Communication in this analysis was conducted using iPerf and hping3 as normal and burst traffic, respectively. At approximately 1.5 seconds after the initial communication with node N8, a massive burst of traffic occurred with full link saturation. This scenario resulted in two devices, N1 and N7, losing communication in the non-optimized SDN environment. An analysis of the communication loss identified that the network becomes overburdened in congestion and that roughly 100.13 seconds after the initiation of the experiment was the time approximation for the event above.

4.2 The Controlled State: End-to-End

Table 1. Percentage improvement of the average end-to-end delay in our optimized framework in regard to non-optimized framework for nodes N1, N2, N4, and N6 communicating with N8 simultaneously.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>N₁</th>
<th>N₂</th>
<th>N₄</th>
<th>N₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>8.29 %</td>
<td>7.86 %</td>
<td>17.13 %</td>
<td>21.85 %</td>
</tr>
<tr>
<td>Median</td>
<td>9.33 %</td>
<td>7.75 %</td>
<td>18.40 %</td>
<td>20.75 %</td>
</tr>
</tbody>
</table>

The results of the evaluation of a non-optimized SDN controller in the previous subsection showed that a network without any optimization lead to communication failure. Next, we examine and analyze the effectiveness of end-to-end delay amongst Traditional Ping, Packet Probing, and Timestamp Recording. Providing a fair comparison, we conducted the same scenario once again, but with one method implemented per execution. A deep investigation of the Traditional Ping, Packet Probing, and Timestamp Recording demonstrated their effectiveness in combating the communication loss. Figure 9 illustrates cumulative distribution
functions (CDFs) of the average results of each end-to-end delay method under the scenario conducted from Section 4.1.

The analysis of the experiments projected that approximately 60 seconds after starting the experiment, the Traditional Ping technique performed as expected during scenarios of massive traffic burst for the situation where nodes N2 and N4 transmitted prioritized traffic to server N8 while N6 communicated normal networking traffic to N8. The Traditional Ping approach showed concerning performance results for node N6 where communication loss occurred due to network congestion as projected by the discontinuity on the graph. Node N1 presented a similar concern for communication loss, but through analysis, the event occurred due to the recalculation of the end-to-end scheme. Moreover, under the same experimental scenario discussed above, Table 1 represents percentage improvements that reflect the reduction in end-to-end delay provided by our optimized SDN framework with regard to a non-optimized SDN environment. The table shows that the proposed SDN-based solution reduces the end-to-end delay from 8.29% to 22% for the different nodes communicating with server N8. Finally, Figure 8 depicts the comparisons of CDF between the non-optimized SDN environment and three delay measurement methods. Table 2 shows the average results of each measurement between each node and server N8. The obtained results demonstrate that our proposed SDN framework can deliver network traffic with significantly lower end-to-end delay in comparison to the non-optimized SDN environment and packet probing.

Table 2. An average end-to-end delay analysis over a period of five minutes under the configuration of mixed communication of prioritized and non-prioritized traffic.

<table>
<thead>
<tr>
<th>Client</th>
<th>Non-Optimize</th>
<th>Ping</th>
<th>Packet Probe</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>4.33 ms</td>
<td>3.37 ms</td>
<td>3.22 ms</td>
<td>3.21 ms</td>
</tr>
<tr>
<td>N2</td>
<td>4.06 ms</td>
<td>2.78 ms</td>
<td>3.05 ms</td>
<td>2.41 ms</td>
</tr>
<tr>
<td>N4</td>
<td>4.29 ms</td>
<td>3.08 ms</td>
<td>3.18 ms</td>
<td>2.57 ms</td>
</tr>
<tr>
<td>N6</td>
<td>2.17 ms</td>
<td>3.19 ms</td>
<td>2.88 ms</td>
<td>2.83 ms</td>
</tr>
</tbody>
</table>

4.3 Examination of Dijkstra’s Algorithm

From the given results in previous subsections, Timestamp Recording appeared to have the least impact (i.e. communication loss and end-to-end delay increase) to burst traffic when an emergency occurred. Additionally, Timestamp Recording was able to minimize end-to-end delay for nodes transmitting emergency data with an overall lower latency as previously described. To investigate in further detail on the approach of minimizing end-to-end delay, we examine Dijkstra’s algorithm during the event of the initial burst of emergency data to see the effects of path selection. Under Figure 11, we see the outcome of Dijkstra algorithm during the initial burst where node N4 communicates with server N8. As described in the figure, three distinguishable paths are depicted (one, two, and three). Table 3 represents the

Figure 9. A comparison of the three approaches where low priority nodes N1 and N6 and high priority nodes N2 and N4 nodes communicate with N8. TS, PP, and Ping represents Timestamp Recording, Packet Probing, and Traditional Ping, respectively.

Figure 10. Client N1 communicated to server N8 for approximately six minutes. A scheduled event executed a massive burst of normal traffic, transmitted from N2, N4, and N7 to the previously mentioned server (N8). The loss of communication, as shown by the discontinuity in the graph, was due to an overly congested link in the infrastructure.
Figure 11. Implementation of Dijkstra’s algorithm from N4 to N8 where we utilize only three paths in the calculation. A visual representation of Dijkstra’s algorithm where D1 and D2 contain an undefined quantity of both emergency and normal traffic and they are being transmitted to server N8. A latency of the network topology prior to N4 communication.

In Table 3, L1, and L8 are unknown latency measurements from an end-user device to an OVS device. Additionally, client nodes attached to L1 and L8 are considered edge nodes and therefore, communication is limited to the sole path (edge link) provided. As depicted, the transmission and return latency are of the same value, and this is due to the data type we transmitted (TCP). To better understand the behavior of Dijkstra’s algorithm, we increase the amount of traffic transmitted to 5GB of UDP data.

Although, using a combination of TCP and UDP did not appear to influence the given network topology links in comparison to pure TCP-based communication. D1 showed subtle latency differences where the transmitting and receiving latencies were 17ms and 22ms, respectively. Additionally, because there were minimal differences with regards to the directionality of traffic, the calculation of Dijkstra had a similar setup. Lastly, even though we only utilize UDP traffic as a one direction method to examine latency per link, normal switch management traffic such as LLDP, BDP, ARP, and other various protocols were still present within our configured topology.

4.4 Quality-of-Service Examination

The end-to-end delay and traffic prioritization are key components to QoS delivery. There are numerous ways to provide QoS for a network topology ranging from the Class of Service (CoS) to the examination of the source and destination IP and MAC addresses. For our brief QoS analysis, we utilize source addresses, destination addresses, and SDN switch ports as our method to identify traffic for priority consideration. The following describes the parameters of our QoS analysis:

Source Address Matching: The source address per SDN flow should match a high priority traffic-based node. Nodes may send a combination of high and low-priority traffic, and to distinguish a difference, further matching is necessary. The result of using a single approach is that all traffic would be considered high priority.

Destination Address Matching: Given that network communication may have a particular prioritization, depending on QoS configurations, traffic destined to a particular system can be feasibly prioritized based on the destination address field per an SDN flow.

Table 3. Delay measurements using Timestamp Recording where node N4 is transmitting traffic to server N8 during an emergency event. L1 and L8 are stated UNKNOWN as they represent edge links in our ExoGENI topology. We are unable to determine the latency of edge links as the SDN Controller has no communication to end nodes (i.e., user devices). Lastly, the top three paths for Dijkstra were calculated and expressed at the bottom of the table.

<table>
<thead>
<tr>
<th>Link</th>
<th>Transmit Latency</th>
<th>Return Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>UNKNOWN</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>L2</td>
<td>52ms</td>
<td>52ms</td>
</tr>
<tr>
<td>L3</td>
<td>39ms</td>
<td>39ms</td>
</tr>
<tr>
<td>L4</td>
<td>25ms</td>
<td>25ms</td>
</tr>
<tr>
<td>L5</td>
<td>5ms</td>
<td>5ms</td>
</tr>
<tr>
<td>L6</td>
<td>15ms</td>
<td>15ms</td>
</tr>
<tr>
<td>L7</td>
<td>13ms</td>
<td>13ms</td>
</tr>
<tr>
<td>L8</td>
<td>UNKNOWN</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>L3,4,7</td>
<td>77ms</td>
<td>77ms</td>
</tr>
<tr>
<td>L2,7</td>
<td>65ms</td>
<td>65ms</td>
</tr>
<tr>
<td>L3,5,6</td>
<td>59ms</td>
<td>59ms</td>
</tr>
</tbody>
</table>

Figure 12. N1 and N2 communicating to N8 where N2 transmitted a massive burst of prioritized traffic and N1 transmitted normal traffic.
SDN Switch Port Matching: Using switch port matching as a way to identify traffic provides an additional verification process in determining that a particular network communication truly originates from a prioritized node. Given a network port or interface that is known to only have a critical system, high valued communication nodes, or an infrastructure dependent resource, a simple QoS policy can be inserted to the network scheme to elevate the traffic communication.

Figure 12 presents our QoS examination for the Timestamp Recording method where N2, the high priority traffic node, was able to ascertain and maintain 30 ms less end-to-end delay in comparison to node N1 that transmitted lower priority traffic.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a timely SDN-based framework for handling network traffic in end-to-end communication systems that are typical of prioritized application services in smart cities. As the end-to-end delay in the network environment plays a key role in the successful delivery of such services, we have presented efficient schemes that identify routing paths with optimal (minimal) end-to-end delay and prioritize network traffic depending on the priority of data transmitted. To better measure the end-to-end delay within end-to-end entities, we carefully investigated and examined the three techniques of calculating end-to-end delay and examined the three techniques of calculating end-to-end delay within end-to-end entities, we carefully investigated and examined the three techniques of calculating end-to-end delay and examined the three techniques of calculating end-to-end delay within end-to-end entities, we carefully investigated and examined the three techniques of calculating end-to-end delay and examined the three techniques of calculating end-to-end delay with a variety of QoS policies such as identifying the traffic of varied priority and address critical security concerns that might affect the end-to-end QoS discussed in this paper. Moreover, we will expand our proposed approach into a variety of applications in smart cities.

6. ACKNOWLEDGMENTS

We acknowledge National Science Foundation (NSF) to partially sponsor the work under grants #1633978, #1620871, #1620862, #1620868, and BBN/GPO project #1936 via an NSF/CNS grant. We also thank FC2 for its seed grant. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of NSF.

7. REFERENCES


ABOUT THE AUTHORS:

Tommy Chin received a B.S. degree in Applied Networking and System Administration in 2013 and is currently pursuing an M.S. degree in Computing Security at Rochester Institute of Technology. Tommy holds numerous industry level certifications, awards, and achievements. His current research focuses on Software-Defined Networking (SDN), Wireless Sensor Networks (WSN), and network security.

Mohamed Rahouti received an M.S. degree in Statistics and is currently pursuing a Ph.D. degree in Electrical Engineering at the University of South Florida. His current research focuses on computer networking and security with applications to smart cities.

Kaiqi Xiong is an Associate Professor at the University of South Florida. He received his Ph.D. degree in Computer Science from the North Carolina State University. Before returning to academia, he had worked in IT industry for several years. He received the Best Demo Award at the 22nd GENI Engineering Conference (GEC22) and the US Ignite Application Summit with his team in 2015. National Science Foundation (NSF), NSF/BBN, Air Force Research Laboratory (AFRL), Amazon AWS, Florida Center for Cybersecurity (FC2), and Office of Naval Research (ONR) have recently supported my research. His research includes security, networking, and data analytics with applications cyber-physical systems, cloud computing, sensor networks, and the Internet of Things.
ABSTRACT
This paper aims at resolving the issues related to increased dimensionality of data in data mining. In this paper, Sparse Cholesky decomposition (SCD) is combined with Non-integer Matrix Factorization (NMF) to remove the problems arising due to increased data dimensionality. The increased data dimensionality in datasets is probably due to non-orthogonality of datasets. The complex conjugate values is used to remove the sparse matrix and a complex gradient algorithm reduces the sparse matrix by the extraction of conjugate values. The SCD-MNF extracts the feature vector and upper triangular matrix linearly maps the feature vector obtained from the SCD. Hence, NMF is employed with SCD for structuring the datasets and this helps to form a well-defined data geometry. The proposed system is evaluated against normalized mutual information and accuracy against different text datasets. The results prove that SCD-NMF attains better results than conventional methods in finding the instances related to the given query.

CCS Concepts
• Information systems → Collaborative filtering;

Keywords
Sparse Cholesky decomposition; Non-integer matrix factorization; Defined Geometric Structure; Conjugate gradient algorithm; Sparse matrix

1. INTRODUCTION
In recent times, larger datasets poses a serious challenge in data processing over several fields. It occurs mainly due to the presence of incomplete or redundant information in large datasets. Hence, extraction of knowledge from such datasets is inaccurate using data mining tools. In order to solve such critical issues in data processing, dataset reduction is required to improve the process of data mining. This creates a higher impact in increasing the data processing capability associated with large datasets [1]

The data reduction while processing the data in abundant amount, reduces the dimensionality of datasets by retaining the data representation. Proper feature selection provides better way to reduce the incomplete datasets. The feature selection is attained by data mining algorithm but it suffers from in-compatibility issues related to feature classification. Such in-compatibility arises due to redundant information available in larger datasets. Hence, the computational complexity of system increases [2] as the data features increases and this leads to increased scalability issues. Further, this leads to reduced accuracy and increased storage requirements. The other difficulties associated with increased dimensionality of data include improper association of variables in feature space, incapability of handling larger data, poor classification and improper knowledge generation.

Several matrix factorization methods are employed to represent the data using its data capturing capability [17,18]. The well-known techniques for matrix factorization includes Singular Value Decomposition (SVD) [19], Nonnegative Matrix Factorization [20] and Principal Component Analysis (PCA) [23]. These methods fail to process high dimensional dataset as it is very compact with low dimensional data representation for learning and visualization.

Non-integer Matrix Factorization is an unsupervised learning method which decomposes a nonnegative data matrix into two nonnegative factors. This is used to represent a new form of original data. The part based data representation using matrix factorization is applied over several application. There are several variations of NMF, which is used to provide customized data representation. The unsupervised methods include Graph regularized NMF (GNMF), Dual GNMF [21], Multiple GNMF [22]. There are other NMF approaches, which include semi-supervised NMF introduced data point labels embedded with NMF objective function.

There are various methods employed with Cholesky decomposition, which is used to eliminate the incomplete or redundant elements in larger data sets. This includes pivoted Cholesky decomposition with low rank approximation [6], quaternion Cholesky decomposition for preserving the structure of data [7], Cholesky decomposition with evolutionary power spectral density for bringing the coherency in data [8], nested Cholesky decomposition to estimate the co-variance matrix [9], Cholesky decomposition to test the incomplete data [10], positive semi-definite matrix based Cholesky decomposition [11], non-singular correlation ma-
The core drawback in semi-supervised NMF is the poor reduction of data points on available data points. In this paper, we proposed Sparse Cholesky Decomposition based NMF to resolve the data dimensionality associated with large scale datasets. This increases the discoverability and training ability over large datasets. Thus, the proposed method resolves the problems in larger feature space using NMF problem. Further, the novelty of study is improved by removing the incomplete datasets which is out carried by Sparse Cholesky Decomposition (SCD). This method uses graph based approach to eliminate the problems of non-orthogonality in NMF. This method resolves the poor decomposition problem with its graph based SCD that avoids rapid multiplication by gradients in feature space.

The outline of the paper is presented as follows: The conventional Cholesky decomposition variants are shown in Section 2. Section 3 provides the improved NMF and section 4 discusses the improvement of NMF using SCD. Section 5 evaluates the proposed method against three datasets to prove its accuracy. Finally, section 6 concludes the paper.

2. RELATED WORKS

There are different types of Cholesky decomposition employed in various applications to improve its functionality. Pivoted Cholesky decomposition with low rank approximation in [6] is intended to compute low-rank approximations of positive dense semi-definite matrices. The trace norm is further used to control the truncation error arising from low rank approximation. This method exhibits fast exponential decay by providing exponential convergence rate under eigenvalue matrix. //

Quaternion Cholesky decomposition [7] is used to preserve the data structure using Hermitian positive definite matrix. This matrix is applied on real matrix representation such that the computational complexity of algorithm is reduced. Cholesky decomposition [8] with evolutionary power spectral density for bringing the coherency in data. This decomposition is applied for power spectral density (PSD) or evolutionary power spectral density matrix and it decomposed further into lagged coherence matrix.

Nested Cholesky decomposition [9] is used to estimate the covariance structure in multivariate longitudinal data analysis. The entries in the nested Cholesky decomposition is interpreted as innovation covariance matrices and moving average coefficient matrix. The elements are modelled as unconstraint linear class model and a Fisher scoring algorithm is developed to measure the regression parameters using maximum likelihood estimator. This enables the consistency of estimators using this method. Finally, the relevant variables are selected using smoothly clipped absolute deviation penalty. Cholesky decomposition to test the incomplete data [10] in the form of conditional independent normal model. This method derives the expressions for unbiased estimators and maximum likelihood estimators and finally best equivariant estimators are introduced by a special group.

Positive semidefinite matrix based Cholesky decomposition [11] is used to identify the zero row or column position and nonsingular submatrix is used to choose these positions with maximal rank. Nonsingular correlation matrix based Cholesky decomposition [12] is proposed based on semi-partial correlation coefficients and equivalent form of the square roots of the differences between two ratios of successive determinants. This method offers an alternative to multiplicative partial correlation and spherical factorization Cholesky matrix. Incomplete Cholesky decomposition [13] is proposed to build a low rank approximation for covariance matrix. This is intended to increase the accuracy with reduced storage using arithmetic operation. This method combines low rank approach with sparse approximation over covariance matrix.

Semi-parametric Cholesky decomposition [14] brings dependence in longitudinal data. This method uses profile kernel approach using semiparametric partially linear regression model for covariance structure. This method provides high data correlation with improved accuracy. QR Cholesky decompositions [15] uses fast moving window algorithm, which updates and downdates the data. This method uses chambers approach for downdating and the procedure is improved with inner products. Detailed analysis is carried out to add or delete a row in original matrix and it provides its superiority over other algorithms. Incomplete Cholesky decomposition [16] is used with Gram-A^S-Schmidt orthonormalization to overcome the computational burden in large training datasets. This method formulates generalized discriminant analysis as another eigenvalue problem of smaller matrix than kernel matrix. This method finds optimal discriminant vector over the training sample and improved the recognition accuracy over large datasets.

3. IMPROVED NMF METHOD

NMF is considered as a low-rank approximation method with non-negative integers, which consists of certain limitations in feature space occurring due to the presence of non-negative matrices in feature vector. Non-negative matrices leads to non-orthogonality problem. Hence, the proposed method uses NMF method and adopts various improvements to reduce the presence of non-negative matrices. Such intuitive method is divided into two parts for representing the data elements.

1. SCD representation of Data elements
2. NMF representation of Feature vector

3.1 Fitness Function for NMF

Since, non-negative elements are present in given matrices, NMF approximates the factorization process. Input data matrix X = (x₁, x₂, ...., xn) of input data vectors of n elements and then the input data matrices are split into two
using decomposition principle using Eq.(1)

\[ X \approx FG^T \]  \hspace{1cm} (1)

where \( X \in \mathbb{R}^{p \times n} \), \( G \in \mathbb{R}^{n \times k} \), \( F \in \mathbb{R}^{p \times k} \) and \( \mathbb{R} \) is a non-zero real number set and \( G = (g_1, g_2, g_3, ..., g_n) \) and \( F = (f_1, f_2, ..., f_n) \). Generally, the value of \( p \) is always lesser than \( n \) and the rank of matrices \( F \) and \( G \) is always lesser than the value of \( X \) i.e. \( k \ll \min(p, n) \). Hence, rank of \( G \) and \( F \) matrices is formed with minimum fitness function. The fitness function is evaluated using squared errors sum, which is represented in Eq.(2)

\[ \min_{F,G \geq 0} J_{sse} = \| X - FG^T \|^2 \]  \hspace{1cm} (2)

Frobenius norm is used to find the matrix normalization and the value of matrices \( F \) and \( G \) found to be non-negative in its Euclidean space with non-orthogonal vectors. Thus, I-divergence function is used to generate non-deficiency cases for rank of matrices \( R \) and \( G \), which is given in Eq.(3),

\[ \min_{F,G \geq 0} J_{ID} = \sum_{i,j=1}^{n} \{ X_{ij} \log \frac{X_{ij}}{FG^T_{ij}} - X_{ij} + (FG^T)_{ij} \} \]  \hspace{1cm} (3)

The Eq.(3) depends on the following cases, when the value of \( I(x) = x \log x - x + 1 \geq 0 \), then the condition is said to have hold the inequality constraint i.e. \( x \geq 0 \). On the other hand, if the value of \( x = 1 \), condition is said to have hold the equality constraint. Thus the inequality condition with I-divergence is expressed as: \( I(u,v) = 1 - u/v + (u/v) \log(u/v) \).

### 3.2 NMF Clustering

The initialization of NMF is considered as an initial step in clustering process but with minimal fitness function always under go local minimum problem [3]. Therefore the intrinsic function is said to have non-convex condition even if the fitness function is convex. The matrices are randomized during initialization of factor matrices but this is ineffective, since local minima is attained due to slow convergence. To avoid such discrepancy, the proposed method uses document clustering approach with NMF. Finally the text documents are clustered using NMF principle [4]. Thus the individual instances related to the given query is assigned over each cluster and feature representation \( g \) is hence considered maximum, which is given by the Eq.(4).

\[ c_g = \arg \max_c g_c \]  \hspace{1cm} (4)

where \( g_c \) is the \( c^{th} \) element of \( g \).

### 3.3 Representation Learning using NMF

The representative learning \( G \) uses supervised method with NMF to reduce the data dimensionality in an effective way. Euclidian space is used for representative learning on \( G \). This method does not take into account the non-orthogonality problem and hence the SCD is used to reduce the larger data dimensionality in feature space.

### 4. SPARSE CHOLESKY DECOMPOSITION

When Cholesky factorization solves large sparse symmetry of linear equation, nonzero elements are formed when matrix is said to have zero elements. This is quite serious in large database and sparse Cholesky factorization considers nonzero sparse matrix structure for finding nonzero Cholesky factor. Hence, elimination graph sequences are used to model the combinatorial problem. This problem is modeled using SCD using matrix and elimination graph.

Consider a sparse matrix \( A \) with zero element at the position \( A(i,k) \) and matrix factorization is used to turn the zero element into non zero element \( L(i,k) \). The nonzero element \( L(i,k) \) with \( k \leq i \leq n - 1 \) is generated into two conditions. Initially, nonzero element \( A(i,k) \) has sparsity pattern of lower triangular matrix of \( A \) over \( L \). Finally, second case when \( A(i,k) = 0 \) with column \( j \) when \( 0 \leq j \leq k - 1 \) such that \( L(i,j) \neq 0 \) and \( L(k,j) \neq 0 \).

The nonzero structure in Cholesky factor \( L \) is determined using graph model, where the matrix position \( A(i,k) = 0 \) is transformed to graph \( G \). The graph \( G \) does not have any edges between its vertex in place of row \( i \) and \( k \). When a vertex having row \( j \) adjacent to graph \( G \) with row \( i \) and \( k \) vertices, then it forms new edge between the two vertices and this helps to remove the row \( j \). This process is repeated for all neighbors of row \( j \) and hence the vertex elimination rule is applied over the rows.

Consider a graph \( G \) with vertex \( v \), which generates a graph \( G \) and this adds edges. This forms the vertices adjacent to \( v \) and these adjacent vertices are considered pairwise adjacent and (ii) further it removes the vertex \( v \) and its instance edges. The graph \( Gv = V \setminus \{v\}, E(V \setminus \{v\}) \cup D(v) \) is thus considered as elimination graph of graph \( G \). Elimination process in ordered graph \( G_n = (V, E, \alpha) \) is \( P(G_n) = G_0, G_1, ..., G_{n-1} \), where the elimination graph is a recursive function of \( G_0 = G \) and \( G_{i+1} = G_{n}^{(i)} \) for \( i = 0, 1, ..., n - 2 \).

During representative learning, the non-orthogonality problem over graph \( G \) occurs due to distance (squared) formulation between the instances \((g_i, g_j)\), which are paired. Hence, the orthogonality is estimated as \((g_i - g_j)^T (g_i - g_j)\). The squared distance assumes implicitly that \( g_i \) deceit in Euclidian space. Generally, feature learning \((f_1, ..., f_n)\) with NMF is non-orthogonal and squared Euclidean distance is considered inappropriate at representative learning by \( G \). Mahalanobis distance (M) measurement is used to solve the inappropriate cases and further it estimates the generalized squared distance metric, thus solving non-orthogonality issues arising due to feature vector, which is given in Eq.(5).

\[ (g_i - g_j)^T M (g_i - g_j) \]  \hspace{1cm} (5)

The properties of NMF is decomposed into data matrix, where \( X \) into,

1. \( F \) with column vectors \((f_1, f_2, ..., f_n)\) spans the feature space of the matrices,
2. The feature space representation is provided by \( G \).
This decomposition property further exploits the SCD-NMF into

1. NMF metrics are estimated using trained feature vectors.
2. SCD finds the upper triangular matrix.
3. Linear data vectors are mapped using upper triangular matrix.

4.1 NMF Metric Estimation
The data matrix \((X)\) in NMF is approximated and \(G\) is the feature space data representation and \(F\) is the data space feature representation. The \(f\) is normalized into \(f^T f = 1\) with the matrix \(M\). Thus the feature vector with gram matrix is estimated using FGF

\[
M = F^T F \mid u^T_{(l)} u, \forall l = 1, 2, \ldots q
\]  

(6)

4.2 Sparse Cholesky Decomposition over NMF metric
The estimation of metric using Eq.(5) assures the approximation of symmetric semi-definite positive matrix \(M\). Further, \(M\) is guaranteed with Linear algebra and hence SCD decomposes the upper triangular matrix \(T\) as shown in Eq.(7)

\[
M = T^T T
\]  

(7)

The eq.(7) replaced with eq.(5) to form the SCD function. Thus the upper triangular matrix \(T\) representation is given by eq.(8)

\[
G \rightarrow TG
\]  

(8)

Algorithm 1 SCD-NMF

\begin{align*}
SCD - NMF(X, NMF, q, parameters) \\
1. & \text{Find } X \in \mathbb{R}^{n \times n}, NMF, qandparameter(NMF) \\
2. & \text{F, G := runNMFonXwithparameterandq (metric estimation)} \\
3. & \text{M := } F^T F \\
4. & \text{T := CD(M) | M = } T^T T \\
5. & \text{Apply CF once the linear coordinates changes, } x := TGy \text{ and det } TG \neq 0 \\
6. & \text{Set } x_t := TG^{-1} y_t \\
7. & \text{Set the preconditioner } M := (TG^T TG)^{-1} \\
8. & \text{Multiply } TG \text{ by } TG^{-1} \\
9. & \text{Compute } x_t := TG^{-1} y_t \\
10. & \text{Return } M, x_t, TG
\end{align*}

This algorithm slows the steps of multiplication and hence it increases the rate of convergence. Once the multiplication process \(TG\) with \(TG^{-1}\) is completed, \(x_t = TG^{-1} y_t\) is computed and reordered after it gets multiplied with the SCD metrics \(M\).

5. EXPERIMENTAL RESULTS
In the proposed method, the documents are clustered and effectiveness is proved against various state of art methods. It includes NMF, GNMF, NPNMF, MM-NMF, RNMF and cd-NMF. The proposed method is evaluated against various text data sets, which includes: Reuters 21578 data, 20 Newsgroups data, and R52 data.

5.1 Evaluation and Comparisons
In the evaluation process, NMF is considered as reference point algorithm. The GNMF preserves the data geometry using KNN graph. NPNMF uses graph approach with linear embedding operation and trained regularization term. MMNMF explores multiple data structure using graph approach. RNMF encodes the geometric structure of data using hyper graph matrix factorization. Finally, the cd-NMF technique uses Cholesky decomposition to eliminate the non-negative matrices and structures the data. The entire proposed system is compared against these previous methods against NMI and accuracy over the testing datasets.

5.1.1 Text Mining Datasets
The proposed Sparse Cholesky Decomposition based NMF is tested over three text datasets, namely, 20 Newsgroups dataset, Reuters 21578 dataset and R52 data. The 20 Newsgroups dataset attributes is shown in Table 1, Reuters 21578 dataset attributes is shown in Table 2 and R52 dataset attributes is shown in Table 3. The sub-clusters selection for the sample formation from the three datasets are shown in Table 4.

<table>
<thead>
<tr>
<th>Class</th>
<th>Total no. of Documents</th>
<th>Total Train Documents</th>
<th>No.of Test Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>soc.religion.christian</td>
<td>996</td>
<td>598</td>
<td>398</td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>990</td>
<td>545</td>
<td>364</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>940</td>
<td>564</td>
<td>376</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>775</td>
<td>465</td>
<td>310</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>628</td>
<td>377</td>
<td>251</td>
</tr>
</tbody>
</table>

5.1.2 Clustering Metrics
The performance of proposed method is estimated using result metrics, namely, Normalized Mutual Information (NMI) and accuracy. The accuracy is used to find the overall performance of proposed system over the given dataset and it is defined as the fraction of correctly clustered data samples and overall data samples.

\[
\text{Accuracy} = \frac{\eta_i}{\eta_{ov}}
\]  

(9)

where, \(\eta_i\) is the data samples which are clustered correctly and \(\eta_{ov}\) is the entire data samples available in the dataset. The
Table 2: Reuters 21578 dataset Attributes

<table>
<thead>
<tr>
<th>Topics</th>
<th>Total Train Documents</th>
<th>Total Test Documents</th>
<th>Other Documents</th>
<th>Total Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1828</td>
<td>280</td>
<td>8103</td>
<td>10211</td>
</tr>
<tr>
<td>1</td>
<td>6552</td>
<td>2581</td>
<td>361</td>
<td>9494</td>
</tr>
<tr>
<td>2</td>
<td>890</td>
<td>309</td>
<td>135</td>
<td>1334</td>
</tr>
<tr>
<td>3</td>
<td>191</td>
<td>64</td>
<td>55</td>
<td>310</td>
</tr>
<tr>
<td>4</td>
<td>62</td>
<td>32</td>
<td>10</td>
<td>104</td>
</tr>
</tbody>
</table>

Table 3: R52 dataset Attributes

<table>
<thead>
<tr>
<th>Class</th>
<th>No. of Train Documents</th>
<th>No. of Test Documents</th>
<th>Total Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crude</td>
<td>253</td>
<td>121</td>
<td>374</td>
</tr>
<tr>
<td>Earn</td>
<td>2840</td>
<td>1083</td>
<td>3923</td>
</tr>
<tr>
<td>Interest</td>
<td>190</td>
<td>81</td>
<td>271</td>
</tr>
<tr>
<td>Money-supply</td>
<td>123</td>
<td>28</td>
<td>151</td>
</tr>
<tr>
<td>Trade</td>
<td>251</td>
<td>75</td>
<td>326</td>
</tr>
</tbody>
</table>

Table 4: Selection of Dataset Samples

<table>
<thead>
<tr>
<th>Samples</th>
<th>20 News Group</th>
<th>Reuters 21578</th>
<th>R52</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>#2</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>#3</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>#4</td>
<td>8</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>#5</td>
<td>8</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>#6</td>
<td>7</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>#7</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>#8</td>
<td>5</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>#9</td>
<td>5</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>#10</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>#11</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>#12</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>#13</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>#14</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>#15</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>#16</td>
<td>0</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>#17</td>
<td>15</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>#18</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>#19</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>#20</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Mutual Information ($MI$) is utilized to calculate the interdependency between the text variables (in this case), which is defined in Eq.(10)

$$MI(x, y) = \sum_{\hat{y} \in Y} \sum_{\hat{x} \in X} p(x, y) \log \frac{p(\hat{x}, \hat{y})}{p(\hat{x})p(\hat{y})}$$ (10)

where $p(\hat{x}, \hat{y})$ is the joint Probability Distribution Function (PDF) of $x$ and $y$, $p(\hat{x})$ is the marginal PDF of $x$, and $p(\hat{y})$ is the marginal PDF of $y$.

The Mutual Information ($MI$) defines the total amount of uncertainty associated with the text document $x$ and $y$, where one of the document is useful for reducing the uncertainty associated with other respective document. The Normalized Mutual Information $NMI$ is thus defined in Eq.(11)

$$NMI(x, y) = \frac{MI(x, y)}{\max(E(x), E(y))}$$ (11)

where, $E(x)$ defines the entropy of the text document $x$ and $E(y)$ defines the entropy of the text document $y$.

Figure 1: Accuracy over Newsgroup

The Figures 1-3 shows the accuracy of SCD-NMF against other conventional methods in terms of the three datasets. The results shows that proposed SCD-NMF attains higher accuracy rate than the conventional schemes. The SCD-NMF accuracy gradually increases sample 1 to sample 20 like other conventional methods, however, the accuracy rate...
is higher for SCD-NMF. It is seen that, as the sample dataset is increasing, the accuracy also tends to remain the same or it increases due to removal of non-negative matrices. However, accuracy reduces when the documents are equally clustered. The average accuracy of SCD-NMF is shown in Figures 7-9. The results shows that SCD-NMF is higher than other methods like MM-NMF, PNP-NMF and NMF. However, the average accuracy of SCD-NMF is only slightly higher than cd-NMF, HNMF and GNMF. The average accuracy value of NMI claims that interdependence between documents of same clusters during dataset testing is high. The interdependence of proposed SCD-NMF is higher than other conventional methods over sample datasets. Likewise, the NMI values of SCD-NMF is higher than other conventional methods as seen in Figures 4-6. The average NMI values SCD-NMF is also higher than other methods, which is evident from Figures 10-12.

6. CONCLUSION
In this paper, a SCD is employed over NMF to eliminate the non-orthogonality in non-negative matrix factorization.
This NMF SCD method assembles data vector and eliminates non-orthogonal indices arising from local representation of Non-negative matrix factorization. The use of upper triangular matrix avoids the presence of Non-negative matrix by mapping with feature vectors. This eliminate sparse matrix with conjugate complex gradients from feature vectors. Such graph based Cholesky decomposition improves well the accuracy and NMI of proposed method than other conventional algorithms.

7. REFERENCES


ABOUT THE AUTHORS:

Dr. Jasem M. Alostad received his Ph.D from University of York, United Kingdom in 2006, MS from the Monmouth University, New Jersey, USA in 1996 and B.S (Computer Science) from Western Kentucky University, KY, USA in 1990. He is currently the Director of Computer and Information Centre in The Public Authority of Applied Education and Training (PAAET), Kuwait. He has more than 10 years of experience in both academics and management. He has authored more than 25 technical research papers published in leading journals and conferences from the ACM, Elsevier, Springer etc. His current research interests include Software Engineering, Data Mining, Data Analysis, Big Data, Cloud Computing, and Internet of Things (IoT).