AN ABSTRACT OF THE DISSERTATION OF
Hassan Hussein Sinky for the degree of Doctor of Philosophy in Computer Science presented on May 11, 2017.

Title: Quality of Experience Assurance Methods in Mobile Heterogeneous Wireless Communication Systems

Abstract approved: ____________________________________________

Bechir Hamdaoui

The proliferation of mobile users and internet content has advanced a plethora of research areas. Among these areas include mobile networks, transport layer protocols, and smart cities. Research shows that global mobile data traffic will increase sevenfold reaching 49 exabytes per month by 2021, most of which will be mobile video content, with a percentage projected to reach up to 78% by 2021. This resulted in efforts to integrate various wireless access technologies for improved performance, increased services and inter-connectivity of end users. The recent growth in data demand has prompted researchers to come up with new wireless techniques (e.g., MIMO, cooperative communication, femtocells, etc.) and develop new technologies (e.g., cognitive radio, LTE, etc.) to be able to meet this high demand. However, mobile user quality of experience (QoE) at the transport layer has largely been overlooked. As users become more mobile they are bound to en-
counter multiple wireless technologies and are thus equipped with multiple network interfaces. In addition, mobile users frequently experience handoffs when associating with wireless networks across a path. The contributions of this dissertation are threefold where our focus mainly encompasses timely delivery of internet content to mobile users through transport and physical layers and network infrastructure solutions.

1. First, we investigate transport layer issues when mobile users encounter handoffs to and from fast and slow links. Specifically, cross-layer techniques are applied to the NewReno variant of the Transmission Control Protocol (TCP) to adapt to network conditions related to mobility. Our analysis shows that cross-layer modifications to TCP allows for less queuing delays, lower round-trip times, improved throughput and minimal packet jitter.

2. Second, we investigate a promising transport protocol known as Multi-Path TCP (MPTCP) which allows for mobile devices to leverage a device’s multiple network interfaces to maintain network connections even when endpoints of the connection change. This allows for connections to remain active during less than ideal scenarios when multiple Wi-Fi networks and cellular base stations are encountered across a mobile user’s path. Default MPTCP congestion control protocols still experience service continuity issues when multiple networks are encountered across a path. A coupled, handoff-based cross-layer assisted, MPTCP congestion control algorithm and framework is proposed and designed to address these issues. Our system model monitors a device’s
received signal strength (RSS) in anticipation of a network handoff and congestion windows are proactively adjusted for a more seamless transition and experience for the end user.

3. Lastly, the integration of the aforementioned wireless access technologies allows large geographic locations to be serviced providing millions of end users with continuous connectivity and optimal QoE. However, the world has seen unprecedented urban population growth over the years. By 2050, it is estimated that 70% of the world’s population will be living in cities. Urban communication networks and content delivery networks have been introduced to leverage these technologies to better service cities and users alike. Content delivery networks are designed to improve overall network performance by bringing data closer to the geographical locations of users. However, traditional, regional content delivery nodes do not suffice for efficient and timely content delivery in large urban communication networks. We propose a fundamental shift to content-centric networks by consolidating these large urban communication networks with standalone edge cloud devices known as cloudlets and introducing geographically distributed content delivery cloudlets (CDC) which store popular Internet content. Advanced cooperative caching techniques are proposed, designed and employed at individual CDCs to push content closer to end users. Our proposed solutions are validated using, LinkNYC, a first-of-its-kind urban communications network aiming to replace all payphones in the five boroughs of New York City (NYC) with kiosk-like structures providing free super fast gigabit Wi-Fi to
everyone. The amalgamation of urban population densities, multiple CDC placements and smarter caching techniques helps exploit the ultimate benefits of a content-centric urban communications network and dramatically improves overall network performance and responsiveness.
Quality of Experience Assurance Methods in Mobile Heterogeneous Wireless Communication Systems

by

Hassan Hussein Sinky

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Presented May 11, 2017
Commencement June 2017

APPROVED:

______________________________
Major Professor, representing Computer Science

______________________________
Director of the School of Electrical Engineering and Computer Science

______________________________
Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

______________________________
Hassan Hussein Sinky, Author
ACKNOWLEDGEMENTS

First and foremost I would like to thank God for giving me the strength in fulfilling my goals and ultimately making what I achieved possible. I would like to give thanks to all those who have supported me in any respect during the completion of my degree. In particular my parents, Hussein and Faykah, my brother, Mohammed, and sister, Tassnym, for their continued support. I would also like to especially thank my wife, Elaf, for her unwavering support during this stressful time. I would not have been able to do it without her. To my beautiful daughter, Kadi, who has made all this hard work worth the time and effort. I would also like to thank the King Abdullah scholarship program and my sponsor, Umm Al-Qura University, for their ongoing financial support during my PhD studies. In my daily work I have been blessed with a great group of fellow colleagues who provided effective feedback and constructive suggestions throughout the course of my research. Lastly, this dissertation would not have been completed without the guidance, encouragement and support of my advisor, Dr. Bechir Hamdaoui.
CONTRIBUTION OF AUTHORS

Research presented in this dissertation was supported in part by the National Science Foundation (NSF) under NSF award CNS-1162296 and Umm Al-Qura University through the King Abdullah scholarship program.

Bassem Khalfi offered guidance through weekly sessions where design choices were made for cooperative content-centric cache replacement policies.

Manuscripts in this dissertation were written with valuable feedback from Dr. Mohsen Guizani and Dr. Ammar Rayes.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Dissertation Organization</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>TCP-CLAH Manuscript</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Related Work</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>TCP Handoff Issues</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Handover Caused Issues</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>BDP Performance Metric and Design</td>
<td>16</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Design</td>
<td>17</td>
</tr>
<tr>
<td>2.5</td>
<td>Testing and Results</td>
<td>20</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Performance and Stability Metrics</td>
<td>20</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Test Topology Configuration</td>
<td>23</td>
</tr>
<tr>
<td>2.6</td>
<td>Conclusion and Future Work</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>CLA-MPTCP Analysis and Framework Manuscript</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>Handoff Induced Issues in Current HetNets</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Transport Layer Analysis</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Multi-Path TCP (MPTCP)</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1</td>
<td>MPTCP Architecture</td>
<td>37</td>
</tr>
<tr>
<td>3.3.2</td>
<td>MPTCP Congestion Control</td>
<td>38</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Drawbacks of MPTCP Congestion Control</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Cross-Layer Assisted Coupled Congestion Control</td>
<td>41</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Design</td>
<td>44</td>
</tr>
<tr>
<td>3.4.2</td>
<td>System Model</td>
<td>46</td>
</tr>
<tr>
<td>3.5</td>
<td>CLA-MPTCP Evaluation</td>
<td>55</td>
</tr>
<tr>
<td>3.6</td>
<td>Conclusion</td>
<td>65</td>
</tr>
<tr>
<td>4</td>
<td>Large Content-Centric Urban Communication Networks Proof of Concept</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>67</td>
</tr>
<tr>
<td>TABLE OF CONTENTS (Continued)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2  Content-Centric Urban Communication Networks                                                                                     69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2.1  Content Delivery Networks                                                                                                         70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2.2  LinkNYC                                                                                                                            71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2.3  Hierarchical Clustering                                                                                                             73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2.4  Content Delivery Cloudlet Placement                                                                                               75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2.5  Borough Analysis                                                                                                                   76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3  LinkNYC Mobile Content Delivery                                                                                                     81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3.1  Design                                                                                                                              82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3.2  Evaluation                                                                                                                          85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.4  Conclusion                                                                                                                             87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5  Large Content-Centric Urban Communication Networks Analysis and Framework Manuscript                                                  89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1  Introduction                                                                                                                          90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2  Content-Centric Urban Communication Networks                                                                 93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2.1  Content Delivery Networks                                                                                                         94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2.2  Content-Centric Networks                                                                                                          95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2.3  LinkNYC Framework                                                                                                                 96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2.4  Borough Analysis and Topology Construction                                                                                  97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3  Content Delivery Cloudlet Selection and Placement                                                                               99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3.1  Population Distribution                                                                                                            100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3.2  CDC Hierarchical Clustering Heuristic                                                                                             101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4  Popularity-Driven and Cooperative Content-Centric Caching                                                                        105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4.1  Popularity-Driven Content Caching                                                                                                 105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4.2  Cloudlet Cooperation for Faster Content Access                                                                                110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4.3  Content Popularity Skewness                                                                                                       114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4.4  Cooperative Content Placement                                                                                                     116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5  Framework Analysis: A LinkNYC Use-Case                                                                                               118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.1  Setup                                                                                                                               118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.2  CDC Neighborhood Size                                                                                                              119</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.3  Impact of Content Popularity                                                                                                      122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.4  Content Dynamicity and Estimation                                                                                                  123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.5  User Mobility                                                                                                                     125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6  Conclusion and Future Directions                                                                                                     126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>6 Conclusion and Future Work</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>Bibliography</td>
<td>130</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Layers of the TCP/IP Network Model</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Test Topology Configuration</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>Average jitter for 10 Mbps send rate</td>
<td>25</td>
</tr>
<tr>
<td>2.3</td>
<td>Average jitter for 15 Mbps send rate</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>Average jitter for 20 Mbps send rate</td>
<td>26</td>
</tr>
<tr>
<td>2.5</td>
<td>Average RTT for 20 Mbps send rate</td>
<td>26</td>
</tr>
<tr>
<td>2.6</td>
<td>Queue size of bottleneck interface for 20 Mbps send rate</td>
<td>26</td>
</tr>
<tr>
<td>2.7</td>
<td>Sender’s congestion window for 20 Mbps send rate</td>
<td>27</td>
</tr>
<tr>
<td>2.8</td>
<td>Average throughput for 20 Mbps send rate</td>
<td>28</td>
</tr>
<tr>
<td>3.1</td>
<td>Transport layer performance measurements related to multiple handoffs</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>MPTCP architecture</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>Subflow SNR behavior between mobile and server</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>Subflows must go through same point of access</td>
<td>45</td>
</tr>
<tr>
<td>3.5</td>
<td>CLA-MPTCP stationary stage flow</td>
<td>49</td>
</tr>
<tr>
<td>3.6</td>
<td>CLA-MPTCP system model</td>
<td>50</td>
</tr>
<tr>
<td>3.7</td>
<td>CLA-MPTCP pre-handoff stage flow</td>
<td>52</td>
</tr>
<tr>
<td>3.8</td>
<td>CLA-MPTCP handoff stage flow</td>
<td>54</td>
</tr>
<tr>
<td>3.9</td>
<td>CLA-MPTCP throughput performance using different $R_0$ values compared to Uncoupled Reno, Coupled, and Olia using traditional handoff scheme with $\Delta_0 = 20$ and $\Delta_2 = 12$ dbm.</td>
<td>56</td>
</tr>
<tr>
<td>3.10</td>
<td>Average RTT, re-transmissions, duplicate ACKs and jitter measurements</td>
<td>57</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>3.11</td>
<td>(a) NS3-DCE Network Topology used for simulations. (b) Throughput loss at handoff as a function of $R_0$.</td>
<td>57</td>
</tr>
<tr>
<td>4.1</td>
<td>Microsoft Azure CDN point of presence locations</td>
<td>71</td>
</tr>
<tr>
<td>4.2</td>
<td>Geographic locations of payphones in NYC</td>
<td>73</td>
</tr>
<tr>
<td>4.3</td>
<td>Brooklyn’s LinkNYC Network</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>Clustering of Brooklyn’s LinkNYC network</td>
<td>77</td>
</tr>
<tr>
<td>4.5</td>
<td>Content-centric LinkNYC performance results</td>
<td>80</td>
</tr>
<tr>
<td>4.6</td>
<td>Mobile content delivery cloudlet architecture.</td>
<td>82</td>
</tr>
<tr>
<td>4.7</td>
<td>Simulation cluster topology.</td>
<td>85</td>
</tr>
<tr>
<td>4.8</td>
<td>Mobile content delivery performance results.</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>Microsoft Azure CDN point of presence locations</td>
<td>95</td>
</tr>
<tr>
<td>5.2</td>
<td>Geographic locations of payphones in NYC</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>LinkNYC network construction using minimum spanning trees</td>
<td>98</td>
</tr>
<tr>
<td>5.4</td>
<td>Daytime versus nighttime population densities</td>
<td>101</td>
</tr>
<tr>
<td>5.5</td>
<td>Borough latency and CDC placement</td>
<td>104</td>
</tr>
<tr>
<td>5.6</td>
<td>(a) and (b) Content-centric LinkNYC performance results comparing LRU to pLFU considering homogeneous and heterogeneous content request intensity and using different clustering techniques. (c) Average latency as the number of CDCs increases. (d) Average latency as the capacity of CDCs increases.</td>
<td>107</td>
</tr>
<tr>
<td>5.7</td>
<td>Virtual CDC content depiction</td>
<td>114</td>
</tr>
<tr>
<td>5.8</td>
<td>Popularity indexes as a function of $s$</td>
<td>115</td>
</tr>
<tr>
<td>5.9</td>
<td>(a) sLFU performance as CDC neighborhood size is increased. (b) Comparing traditional cache replacement policies with pLFU and sLFU considering static content popularity skewness.</td>
<td>122</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.10 (a), (b), and (c)</td>
<td>124</td>
<td></td>
</tr>
</tbody>
</table>

Illustrate content-centric LinkNYC performance results comparing LRU, LFU, pLFU and sLFU considering a dynamic network with periodically fluctuating content interests and popularities where $s$ is changed per community with a range between 0 and 2. (d) The ratio of cache hits to cache misses with respect to $s$. 


<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>59</td>
</tr>
<tr>
<td>4.1</td>
<td>72</td>
</tr>
<tr>
<td>4.2</td>
<td>76</td>
</tr>
<tr>
<td>4.3</td>
<td>78</td>
</tr>
<tr>
<td>4.4</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>103</td>
</tr>
<tr>
<td>5.2</td>
<td>106</td>
</tr>
<tr>
<td>5.3</td>
<td>106</td>
</tr>
<tr>
<td>5.4</td>
<td>120</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>1 TCP adjustments for handoff to low-delay</td>
<td>19</td>
</tr>
<tr>
<td>2 TCP adjustments for handoff to high-delay</td>
<td>20</td>
</tr>
<tr>
<td>3 CLA-MPTCP Stationary Stage</td>
<td>60</td>
</tr>
<tr>
<td>4 CLA-MPTCP Pre-Handoff Stage</td>
<td>61</td>
</tr>
<tr>
<td>5 CLA-MPTCP Handoff Stage</td>
<td>62</td>
</tr>
<tr>
<td>6 Mobile Content Delivery</td>
<td>83</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

The convenient features of wireless technology, such as mobility, portability, and ease of use and deployment, are among the main reasons behind the technology’s tremendous success. With advancements in medium access control (i.e. MAC) and transmission control protocols (i.e. TCP/IP, UDP) users are equipped with incredible technology providing unprecedented timely access to information from the convenience of handheld portable devices. As mobile devices increase in number and become more powerful so does the demand for optimal performance and service quality.

Due to this growing demand for access to communication services anywhere and any time an accelerated development towards the integration of various wireless access technologies has resulted. Heterogeneous wireless systems have been designed to take advantage of these wireless features and provide efficient content delivery to end users. Heterogeneous wireless systems provide coexisting technologies resulting in more services and higher data rates while also providing a global roaming/always connected environment through a diverse range of mobile access networks.

Upon its inception the Transmission Control Protocol (TCP) has largely remained unaltered due to its robust and reliable in-order delivery of data. TCP resides in the transport layer between the application layer and internet layer of
the four layer TCP/IP model. Communication between application programs in
the application layer at a source and destination is done using the transport layer
to reliably transmit data that is routed and sent using the Internet and physical
layers respectively.

Unfortunately, technology is not without its faults. In this evolving industry, a
mobile user’s quality of experience is often times the driving force behind many of
today’s technological research and solutions. We, as users and consumers, generally
take service availability for granted. The smallest disruption or glitch in any of
these technologies would be considered an unbearable inconvenience. Introducing
a degree of mobility to a mobile user’s active connection results in numerous service
continuity and quality of experience (QoE) issues. Within a heterogeneous wireless
system a mobile device may encounter many networks with varying data rates and
performance, sometimes even drastic differences. Such situations negatively impact
user QoE.

A typical heterogeneous networking scenario involves mobile terminals with
multiple network interfaces which are capable of choosing the best possible net-
work or link for data. For instance selecting a Wi-Fi network over a cellular network
when you are in range to save battery and achieve higher data rates. Another example would be having a Skype conversation on a mobile device and seamlessly, gracefully and intelligently select networks in range without the user feeling any disruption during the conversation. This type of graceful network selection is generally referred to as a seamless handoff. A horizontal handoff takes place between points of attachment supporting the same network technology. For instance, a mobile user walking through a university department building may undergo a few horizontal handovers. On the other hand a vertical handoff occurs between points of attachment supporting different network technologies [1]. This occurs if a mobile user exits a building, out of range of Wi-Fi, and switches to a cellular data network. Handoffs can be categorized into two types:

- **Break before make** terminates the old link before the handoff completes. This causes a visible disruption in connectivity and packet losses as packets that have been sent using the old interface will have to be retransmitted.

- **Make before break** terminates the old connection only when the new link is operable. This type allows for no packet loss due to buffering and is referred to as a soft handover.

Drawbacks and issues begin to arise when handovers occur between networks with varying capacities and data rates such as between Wi-Fi and cellular networks [2], [3]. We investigate and analyze transport layer issues due to network handovers and propose cross-layer assisted solutions to improve overall performance by minimizing user perceived *disruptions* and *glitches*. Our measurements show
that our solutions outperform traditional methods significantly in terms of jitter, round-trip time, queue size, throughput convergence and overall stability, yielding much more robust, reliable, handoff resistant and efficient data transfers.

QoE issues are exaggerated in densely populated environments with dynamic user presence and Internet content interests. The world has seen an unprecedented urban population growth over the years. In fact, the number of urban residents has increased by nearly 60 million a year. By 2050, it is estimated that 70% of the world’s population will be living in cities. In addition, research shows that global mobile data traffic will increase sevenfold reaching 49 exabytes per month by 2021, most of which will be mobile video content, with a percentage projected to reach up to 78% by 2021. The aforementioned integration different wireless access technologies allows for large geographic locations to be serviced providing millions of end users with continuous connectivity and optimal quality of experience (QoE). Large urban communication networks and content delivery networks have been introduced to leverage these technologies to better service cities and users alike. Urban communication networks have evolved over the years to address urban challenges through the use of information, communication technology and the Internet. Content delivery networks are designed to improve overall network performance by bringing data closer to the geographical locations of users. Traditionally, content delivery nodes or datacenters are geographically distributed throughout the world servicing different regions. However, in large urban networks the same content may be requested by multiple users resulting in the content traversing the network multiple times, over large distances, to and from a remote content delivery node.
hosting the content. Thus, efficient and responsive content delivery solutions are needed for a more optimal user QoE. In order to address these issues we investigate and propose a fundamental shift of large urban communication networks to content-centric networks. Building such a network infrastructure capable of adequately servicing urban locations has become increasingly difficult due to the sheer number of Internet devices and users (e.g., rapidly increasing numbers of Internet of Things (IoT) devices).

Coupling urban communication networks with content-centric and delivery principles greatly benefit content producers, consumers and the cities they reside in. Improving their infrastructure using practical approaches to provide more reliable and responsive communications can assist in the technology’s overall success. The underlying concept behind content-centric communication networks is to allow a consumer to focus on the desired named content rather than referencing the physical location or named hosts (IP) where that content is stored. This helps eliminate the need to traverse the Internet for content which reduces infrastructure and bandwidth costs while improving network robustness and QoE. This shift is a product of empirical research resulting in the fact that the vast majority of Internet usage involves data being disseminated from a source to multiple users. The potential benefits of a content-centric adoption include in-network caching to reduce congestion, improved delivery speeds, simpler network configuration and network security at the data level [4]. In this dissertation we combine content-centric and content delivery principles to improve the performance, responsiveness and reliability of large urban communications networks. We validate our methods,
solutions and framework using a currently deployed large urban network known as LinkNYC; a first-of-its-kind urban communications network aiming to replace all payphones in the five boroughs of New York City (NYC) with kiosk-like structures providing free super fast gigabit Wi-Fi to everyone. Our framework lays the foundation for a first-of-its-kind content-centric urban communications network with the focus of minimizing latency and leveraging interest-based content caching through popularity learning and cooperative content placement.

1.1 Dissertation Organization

The organization of this dissertation is presented as several manuscripts. Chapter 2 investigates the need for physical and transport cross-layer assistance in TCP and proposes protocol amendments for mobile users. In Chapter 3 Multi-Path TCP (MPTCP), a promising new transport layer protocol for mobile users, is investigated and a coupled MPTCP congestion control framework is proposed for mobile users. Chapter 4 proposes an architectural shift of large urban communication networks to content-centric networks. Finally, in Chapter 5, we propose a complete content-centric urban framework using LinkNYC’s infrastructure for efficient, reliable and responsive content delivery in dynamic environments.
Chapter 2: TCP-CLAH Manuscript


Hassan Sinky and Bechir Hamdaoui
School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon 97331, Email: {sinkyh,hamdaoub}@oregonstate.edu

ABSTRACT

This paper provides a simulation-based performance evaluation of TCP during handovers between networks with varying data rates. Our focus is to analyze and resolve issues, specifically service continuity, with TCP when drastic changes to the point of access network occurs. A cross-layer assisted handoff TCP (TCP-CLAH) is proposed relying on the bandwidth delay product upper bound ($BDP_{UB}$) of the underlying networks to adjust parameters and minimize handoff induced issues. The BDP product is used to evaluate the capacity of a given link at any point in time and thus a path BDP can be obtained. Given the path BDP upper bound of the point of access network along with cross-layer handoff notifications TCP adjusts its congestion window, round-trip time estimations and duplicate acknowledgments
to accommodate for a network handoff. A series of measurements are conducted to analyze and compare the performance of TCP NewReno (TCP-NR) and our proposed TCP-CLAH. Our simulations show that TCP-CLAH outperforms TCP-NR in terms of jitter, round-trip time, queue size and overall TCP stability during a handoff, yielding much more robust, reliable, seamless, handoff resistant and efficient data transfers.

2.1 Introduction

The convenient features of wireless technology, such as mobility, portability, and ease of use and deployment, are the main reasons behind the technology’s tremendous success. Heterogeneous wireless systems have been designed to take advantage of these wireless features. We as users and consumers generally take service availability for granted. We usually don’t worry or think about it because we assume we will always have it. Not only have we taken for granted service availability but also the quality of that service. Even the smallest disruption or glitch in any of these technologies would be considered an unbearable inconvenience. As the number of mobile users increase so does the demand for top notch performance and service quality. Due to this growing demand for access to communication services anywhere and any time an accelerated development towards the integration of various wireless access technologies has resulted.

Heterogeneous wireless systems provide coexisting technologies resulting in more services and higher data rates while also providing a global roaming/ al-
ways connected environment through a diverse range of mobile access networks. Thus, within a heterogeneous wireless system a mobile device may encounter many networks with varying data rates and performance, sometimes even drastic differences. A typical heterogeneous networking scenario involves mobile terminals with multiple network interfaces which are capable of choosing the best possible network or link for data. For instance selecting a wifi network over a cellular network when you are in range to save battery and achieve higher data rates. Another example would be having a Skype conversation on a mobile device and seamlessly, gracefully and intelligently select networks in range without the user feeling any disruption during the conversation. This type of graceful network selection is generally referred to as a seamless handover.

Drawbacks and issues begin to arise when handovers occur between networks with varying capacities and data rates such as between Wifi and cellular networks. In this paper we analyze TCP issues due to network handovers and propose a cross-layer assisted handoff TCP (TCP-CLAH) to improve overall performance by minimizing user perceived glitches. In this work, TCP New Reno (TCP-NR) is compared with our TCP-CLAH scheme and results are evaluated. Our measurements show that TCP-CLAH outperforms TCP-NR significantly in terms of jitter, round-trip time, queue size, throughput convergence and overall stability, yielding much more robust, reliable, handoff resistant and efficient data transfers.

In our research, a cross-layer assisted handoff TCP is proposed in order to alleviate performance issues and glitches that occur during a handoff between networks of varying performance and data rates. A path BDP upper bound is obtained of
the new network and TCP’s congestion window and RTT estimation values are set based on the network handed off to. Our contributions are as follows:

• Simulation based analysis of TCP NewReno during handovers between networks of different performance data rates.

• Adjust TCP parameters to adapt to network handovers

• Establish that reducing congestion window prior to handover is essential for service continuity, minimal disruption and glitches.

• To our knowledge, this work is the first to investigate handover connection disruptions and glitches with TCP NewReno and applies the path bandwidth delay product upper bound of the underlying point of access network on the congestion window drastically improving service continuity during handovers.

The rest of the paper is organized as follows. We begin by discussing related work in Section 2.2. TCP issues as a result of network handovers are discussed in Section 2.3. We then, in Section 2.4, discuss the bandwidth delay product (BDP) performance metric as well as propose a scheme to improve TCP during a handoff. In Section 2.5, we introduce our test topology and perform a series of tests to evaluate and compare the performance of our implementation of the TCP-CLAH with TCP-NR. Finally, we conclude the paper in Section 2.6.
2.2 Related Work

Network handovers have been a well researched subject. However, researchers today have focused mainly on the handover detection and decision process where little has been done to alleviate transport protocol issues resulting from a change in network point of access especially between networks with varying data rates.

In [1] the authors divide a handoff into three phases. The first phase is when TCP goodput increases, the second phase is when TCP goodput decreases and finally a TCP timeout happens and the sending rate slowly increases. TCP is adjusted based on the phase, user’s speed, location and initial congestion window. The authors mostly focus on the physical aspect of the mobile user and handoff decision rather than the transport protocol itself. A soft handover is not considered.

In [2] Freeze TCP (ftcp) when a pending handover is to occur a mobile user sends a zero window advertisement (ZWA) to the sender. The sender then freezes its timers and suspends all transmissions. Once the handover is complete the mobile user sends three successive acknowledgments to the sender to resume transmission. The time before a disconnection occurs is referred to as the warning period. This period is difficult to predict and if predicted too early the sender enters persist mode too early increasing idle time and decreasing throughput. If predicted too late the sender may not receive the ZWA in time causing the congestion window to reduce due to lost packets and in turn decrease throughput.

In [3], [4] the authors have designed a cross-layer assisted TCP solution for handoff situations. It is shown that TCP can benefit greatly by adjusting pa-
rameters according to a network handoff. When a handoff occurs RTT and RTO estimations are reset only until all packets sent before the handoff have been acknowledged. A reduction in congestion window is done only when needed. In addition, the slow start threshold is set to the congestion window which unnecessarily places TCP in the congestion avoidance phase when a handoff occurs. On the other hand, we propose a different adjustment to TCP based on the path bandwidth delay product upper bound as well as a different principle in setting the RTT and RTO estimations. Finally, our proposed scheme uses TCP NewReno whereas authors in these works use TCP SACK.

In the next section we discuss network handovers and their effect on TCP.

2.3 TCP Handoff Issues

Handovers can be a result of preference, service availability, network quality or forced. As mentioned earlier, researchers have mainly focused on the handover decision and detection processes whereas our work will focus on how to improve TCP in handover scenarios and minimize connection disruptions and glitches. There are two types of handovers. A horizontal handover takes place between points of attachment supporting the same network technology. For instance, a mobile user were walking through a department building may undergo a few horizontal handovers. On the other hand a vertical handover occurs between points of attachment supporting different network technologies [5]. This occurs if a mobile user exits a building, out of range of Wifi, and switches to a cellular data network.
Little has been done in the area of analyzing and resolving TCP’s drawbacks during a handoff. Before introducing these drawbacks; handovers can be of two types:

- **Break before make** terminates the old link before the handoff completes. This causes a visible disruption in connectivity and packet losses as packets that have been sent using the old interface will have to be retransmitted.

- **Make before break** terminates the old connection only when the new link is operable. This type allows for no packet loss due to buffering and is referred to as a soft handover.

TCP is optimized for wired networks where packet loss is assumed to be caused by congestion only [6]. However, when we move to wireless scenarios this assumption fails since wireless links experience error rates much larger than wired networks. Thus, if packets are dropped or delayed the TCP congestion control mechanism is initiated to alleviate the problem which reduces throughput drastically. As a result, during a handoff, the main dilemma we are left with is trying to help TCP avoid incorrectly initiating its congestion avoidance and slow start mechanisms. In order to successfully do this TCP must differentiate between packet delay caused by *actual real packet loss and congestion on the current link* and *packet delay simply caused by vertical or horizontal handoff*. 
2.3.1 Handover Caused Issues

There are different TCP problems that arise depending on the type of handoff that occurs [7]. We classify these TCP issues based on two handoff scenarios:

2.3.1.1 High-delay link to low-delay link

- **Packet reordering**: This issue occurs when the source node switches to considerably faster link. Packets sent on the new link overtake packets that have been send on the old link which causes out of order packets to arrive at the destination resulting in duplicate acknowledgements to be sent (du-packs). This is an issue because during a handoff TCP would misinterpret 3 dupacks as being caused by congestion, loss or link error and enters congestion avoidance and retransmit packets incorrectly assumed to be lost when in reality this was caused by a handoff.

- **Inflated retransmission timeout (RTO)**: Naturally, a high-delay link will have a high RTT and RTO value. Since the RTO is updated once in an RTT, when a handoff occurs, the RTO will converge slowly to the new low RTO value. This is an issue because invoking RTO recovery to recover lost packets will take a longer time due to the high RTO value. In other words, in the event of lost packets on a faster link TCP would not be able recover quickly due to the high RTO value of the high-delay link. Since after the handoff we are on a low-delay link we want TCP to be able to recover lost
packets faster using a smaller RTO representing the new faster link.

2.3.1.2 Low-delay link to high-delay link

- **Spurious or false retransmission timeout (RTO):** This issue occurs when we handoff to a slower link. The low-delay link before the handoff has a low RTO value. Packet acknowledgments before the handoff take the high-delay after the handoff. This causes TCP to falsely or spuriously timeout assuming packets have been lost due to a low RTO value from the low-delay link.

- **Link overshoot:** After a handoff to a low-delay link the sender may inject more data onto the link than can be handled resulting in dropped packets. Specifically when TCP is in the slow start phase the congestion window is doubled for every acknowledgement received. If a handoff to a low bandwidth delay product link (BDP) link occurs during the slow-start phase a congestion window increase may result in dropped packets. This is referred to as a slow-start overshoot. TCP may interpret this incorrectly and reset the congestion window which is a performance hit we want to avoid.

In the next section we introduce the use of the bandwidth delay product (BDP) as a means to evaluate network capacity. A cross-layer assisted handoff TCP scheme is then proposed to minimize handoff induced glitches and disruptions in service continuity.
2.4 BDP Performance Metric and Design

We argue that in order to minimize connection glitches and maintain adequate service continuity during a handoff the bandwidth delay product (BDP) of the link must be considered. The BDP refers to the maximum amount of data on a network link at any given time. In other words it represents the maximum amount of bytes that have been sent but not yet received [8]. The BDP is represented by the product of a link’s capacity and its round-trip time. By definition then the path BDP is the product of the path bottleneck throughput, \( R_{\text{min}} \), along the forward path and the path round-trip time, \( RTT_{\text{path}} \) and is represented by the following equation:

\[
BDP_{\text{PATH}} = R_{\text{min}} \times RTT_{\text{path}}
\]  

(2.1)

The path BDP can be expanded as follows:

\[
BDP_{\text{PATH}} = R_{\text{min}} \times \left( \frac{S}{R_1} + \frac{S}{R_2} \ldots \frac{S}{R_n} \right)
\]

\[
+ \frac{S}{R'_1} + \frac{S}{R'_2} \ldots \frac{S}{R'_m} \right)
\]  

(2.2)

Where \( n \) and \( R_i \) are the number of hops and throughput on the forward path and \( m \) and \( R'_i \) are the number of hops and throughput on the backward path. Naturally, the maximum achievable throughput cannot exceed \( R_{\text{min}} \). As shown in [8] the upper bound of the path BDP can be obtained and is expressed by the
following equation.

\[
\text{BDP}_{UB} \leq R_{\text{min}} \times \left( \frac{S}{R_{\text{min}}} + \frac{S}{R'_{\text{min}}} \cdots \frac{S}{R'_{\text{min}}} \right)
\]

(2.3)

Combining terms the following is obtained:

\[
\text{BDP}_{UB} \leq R_{\text{min}} \times \left( n \frac{S}{R_{\text{min}}} + m \frac{S}{R'_{\text{min}}} \right)
\]

(2.4)

Assuming the forward and backward paths are similar in location resulting in near equivalent data rates this can be simplified to:

\[
\text{BDP}_{UB} = S \times N
\]

(2.5)

Where \( N = (n + m) \) and \( S \) is the packet size. Thus the path BDP cannot exceed \( S \times N \) otherwise packets will be queued at the bottleneck [8].

2.4.1 Design

Our goal is to alleviate issues presented in Section 2.3 by proposing modifications to TCP NewReno allowing for robust data transfers and maintain service continuity. Thus, our intention is to minimize slight disruptions and glitches immediately following a handover.
In designing our scheme two assumptions are made:

- We assume a soft handover scenario where a new connection is made before the previous connection is terminated. This allows for no packet loss due to buffering. These handovers involve scenarios where a mobile node is in range of multiple networks.

- We assume a cross-layer notification of a handoff indicating a drastic change in bandwidth. That is a notification communicated to TCP indicated a significant increase or decrease in bandwidth.

TCP-CLAH brings three modifications to TCP NewReno in order to provide a seamless handover:

- Congestion window is set to the path $BDP_{UB}$ obtained using Equation 2.5

- RTT and RTO estimations are reset upon a handoff as if for a new connection

- Duplicate acknowledgments are temporarily disabled during a handoff

A large congestion window during a handoff can be detrimental to service continuity. Maintaining a large congestion window increases the probability of contention, round-trip time, queuing and packet loss which can degrade TCP performance drastically. We argue, by setting the congestion window to exactly the path BDP upper bound, that is the maximum number of bytes that can be in-air (transmitted but not yet received) across the path, guarantees we are transmitting
the maximum amount of bytes while avoiding queuing at the bottleneck. By using the $BDP_{UB}$ we ensure the throughput is not compromised. As mentioned in Section 2.3 we are concerned with two handoff scenarios.

For a high-delay to low-delay network handoff we apply Algorithm 1. The intuition behind this is that if the RTO is set too high TCP may be slow to recover lost packets. If set too low, we risk spurious RTOs where TCP assumes packets are lost when in reality they may simply be on their way. Through testing we found by resetting these values TCP converges quicker to the new network’s RTT and RTO estimations. The congestion window is set to Equation 2.5. Often times this value is less than the current congestion window size. One would think that a transition from a high-delay network to low-delay network involves further increasing the congestion window to fully utilize the link. However, as mentioned earlier, injecting more data than the path BDP causes packets to be queued at the bottleneck. Thus setting the congestion window to the path BDP, albeit less, is sufficient and produces better results. The achieved throughput is not compromised by this action as will be shown in Section 2.5.

**Algorithm 1** TCP adjustments for handoff to low-delay

```
1: if Notification of handoff to low-delay then
2:   reset $RTT$ and $RTO$ estimations
3:   temporarily disable $dupacks$
4:   obtain path $BDP$
5:   set $cwnd = path BDP$
6: end if
```

For low-delay to high-delay network handovers we apply Algorithm 2. Resetting the RTT and RTO estimations to their initial states allows for TCP to relearn
the new network conditions quicker and avoid spurious RTOs. Again setting the
current congestion window to the path BDP avoids a link overshoot. Since we are
moving to a slower network, the three duplicate acknowledgments are not an
issue.

Algorithm 2 TCP adjustments for handoff to high-delay
1: if Notification of handoff to high-delay then
2: reset RTT and RTO estimations
3: obtain path BDP
4: set cwnd = path BDP
5: end if

In Section 2.5 we analyze the results obtained from measuring the aforementioned service continuity and performance factors (i.e. jitter, round trip time, queue size and throughput) conducted on the test topology in Figure 2.1.

2.5 Testing and Results

In this section we discuss metrics measured to compare the performance and sta-
bility of TCP during a handoff. We then discuss the configuration of our test
topology and how the experiments were conducted.

2.5.1 Performance and Stability Metrics

Overall network performance is impacted by many environmental and physical
aspects and limitations. It is crucial to understand what factors play a role in
network performance and stability and what can be done to improve them. We
discuss and measure factors that influence network stability and performance during handovers. The following subsections briefly discuss these factors.

2.5.1.1 Jitter

Jitter, also referred to as packet delay variation (PDV), is the difference in end-to-end delay between packets. Thus, variable delay causes jitter. On the other hand, a network that experiences a constant amount of latency between packets has no jitter [9]. In addition, as a result of congestion a packet may be queued or delayed on a path where there was no queuing or delay for other packets [10]. This in turn causes a variation in latency. In VoIP scenarios high variation in packet delay of an audio stream can result in poor call quality where audio delays and dropouts are experienced. Thus jitter is a major quality of service impairment in real-time applications and is another useful metric to measure network and broadband stability [11]. In this work, jitter is our main factor influencing service continuity, disruption and glitches. We evaluate the average jitter experienced when a handoff occurs.

2.5.1.2 Throughput

Bandwidth constantly fluctuates and can vary depending on the transmission protocol used (i.e. TCP and UDP). If congestion is detected TCP backs off (slows down) and thus influences network bandwidth causing transmission rate fluctua-
tions [12], [13]. It is a reliable metric essential for representing network performance and stability. Thus, TCP is prone to disruption and glitches caused by handovers. In this work, throughput is evaluated to quantify service continuity, disruptions and glitches.

2.5.1.3 Round-Trip Time

A network undergoing long round-trip times usually is the result of congestion, packet loss and queuing delays. Queued packets can cause variations in round-trip times and can not only increase delays but also packet loss. Larger round trip time (RTT) values indicate the average queue length at intermediate nodes is large. Lower RTT values imply less congestion and low average queue lengths. In this work, TCP round-trip times are evaluated to quantify service continuity, disruptions and glitches.

2.5.1.4 Queue Size

Queue size is another factor that affects service continuity during a handover. The more packets queued at the bottleneck longer round-trip times are experienced. It is essential to keep queue size low immediately following a handover to reduce round-trip times and in turn jitter. In this work, queue size is evaluated to quantify service continuity, disruptions and glitches.
2.5.2 Test Topology Configuration

In order to evaluate TCP-NR and TCP-CLAH, the aforementioned issues are analyzed to characterize network performance, stability and service continuity during a handoff. Of particular interest is the minimization of visible connectivity glitches due to handoff. Glitches involve a sudden increase or decrease in round-trip time and queue size which can be characterized by sudden fluctuations in jitter. A sudden change of the point of access network can cause significant glitches in a connection. Thus, trying to minimize these glitches and maintain a smooth transition is what we evaluate and resolve.

To perform our tests the topology in Figure 2.1 is configured in NS3. As illustrated, two networks of varying data rates are created with a common destination node. This can be viewed as a 54 Mbps Wifi network and a 5.5 Mbps cellular network. The maximum achievable throughput in our test topology is 19 Mbps for
Network 1 and 3 Mbps for Network 2. This is understandable since experimental speeds tend to be much less than theoretical link speeds [14].

The topology is configured in a way such that two paths are available from node Mobile (source) to node Dest (destination); that is one path per network. This helps emphasize the influence of handovers on a connection’s stability, performance and service continuity. Our tests have been encouraging and shine a light on the importance of cross-layer TCP adaptability.

Our tests focus on the aforementioned factors that influence connection glitches and service continuity during handoffs. The conducted tests involve a mobile device moving from one network to another, starting in Network 1, every 15 seconds over a 100 second time period. Recall from Section 2.4 we assume a soft-handover where cross-layer notifications inform TCP of a significant change in bandwidth. During this time period the mobile device has established a connection with the destination node and is transferring data (i.e. YouTube, P2P data transfers). The tests are conducted with varying application sending rates; 10, 15 and 20 Mbps.

Figures 2.2, 2.3 and 2.4 compares the jitter experienced by TCP-NR and TCP-CLAH over the connection for each respective sending rates. As illustrated, TCP-CLAH drastically improves jitter and hence disruptions and glitches are reduced during a handoff. TCP-CLAH improves jitter by 350% for an application send rate of 10 Mbps, a 525% improvement for a 15 Mbps send rate, and a 75% for a 20 Mbps send rate. Disabling dupacks when a handoff occurs prevents unnecessary retransmissions which minimizes the variation in packet delay. In addition, resetting the RTT and RTO estimations allows for TCP to relearn the link condi-
tions gracefully as shown in Figure 2.5. Finally, setting the congestion window to the path BDP upper bound when a handoff occurs we prevent rapid queuing at the bottleneck as mentioned in Section 2.4 and illustrated in Figure 2.6 which in turn allows for shorter initial round-trip times. All these factors influence jitter which is an essential metric for characterizing and visualizing handoff induced glitches.

As we can see from our tests TCP-NR resulted in erratic behavior during network handovers. Specifically large fluctuations in jitter, RTT and queue size. This glitch although small can be visible by the user and can disrupt an otherwise problem free connection. By preemptively adjusting the congestion window to the path BDP upper bound when a handoff occurs we not only avoid rapid fluctuations
Figure 2.4: Average jitter for 20 Mbps send rate

Figure 2.5: Average RTT for 20 Mbps send rate

Figure 2.6: Queue size of bottleneck interface for 20 Mbps send rate
in jitter but also drastically minimize connection disruptions and glitches without compromising throughput. Upon a handoff, the congestion window is much larger than the path $BDP_{UB}$ as shown in Figure 2.7. This is expected as the path $BDP_{UB}$ refers to the maximum number of bytes that can be transmitted in air without being queued.

From Figure 2.8 we can see that throughput is not compromised as a result of setting the congestion window to the $BDP_{UB}$. In fact, upon further inspection immediately following a handoff, TCP throughput arrives quicker to the achievable throughput for both handoff scenarios. Our promising results validate that the BDP is an essential metric that must be considered by TCP in order to minimize handoff induced disruptions and glitches over a connection.

2.6 Conclusion and Future Work

In this work we analyze TCP’s drawbacks as a result of handovers between networks with varying data rates. A list of issues are categorized and presented. A
modification to TCP NewReno is proposed based on the bandwidth delay product of the underlying networks. Our proposed cross-layer assisted handoff TCP (TCP-CLAH) is compared to TCP NewReno. Tests were performed to analyze and evaluate service continuity and emphasize the importance of incorporating the bandwidth-delay product within TCP. Our simulations show that TCP-CLAH outperforms TCP-NR significantly in terms of jitter, round trip time, queue size and stability providing improved performance in handoff scenarios drastically reducing disruptions and glitches. Finally, as future work, we look to extend TCP-CLAH to support multiple TCP flows and multimedia data.
Chapter 3: CLA-MPTCP Analysis and Framework Manuscript

Proactive Multi-Path TCP for Seamless Handoff in Heterogeneous Wireless Access Networks

Hassan Sinky*, Bechir Hamdaoui*, and Mohsen Guizani†

*School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon 97331, Email: sinkyh,hamdaoub@oregonstate.edu
†Department of Electrical and Computer Engineering, University of Idaho, Moscow, Idaho 83844, Email: mguizani@ieee.org

ABSTRACT

Multi-Path TCP (MPTCP) is a new evolution of TCP that enables a single MPTCP connection to use multiple TCP subflows transparently to applications. Each subflow runs independently allowing the connection to be maintained if endpoints change; essential in a dynamic network. Differentiating between congestion delay and delay due to handoffs is an important distinction overlooked by transport layer protocols. Protocol modifications are needed to alleviate handoff induced issues in a growing mobile culture. In this article, findings are presented on transport layer handoff issues in currently deployed networks. MPTCP as a potential
solution to addressing handoff- and mobility-related service continuity issues is discussed. Finally, a handoff-aware cross-layer assisted MPTCP (CLA-MPTCP) congestion control algorithm is designed and evaluated.

3.1 Introduction

The demand for access to information has grown exponentially throughout the years. This accelerated growth has researchers focusing on integrating various wireless access technologies to provide higher data rates, more services and a global roaming culture. Naturally, in heterogeneous wireless networks (HetNets), mobile terminals are bound to undergo multiple handoffs and are equipped with multiple network interfaces to utilize the technologies they encounter. Handoffs can occur horizontally between networks of the same technology using the same interface or vertically between networks of different technologies using multiple interfaces. Due to the improvements in technology and overall power of mobile devices users are put in more mobile situations. As users become more mobile, drawbacks with TCP and other traditional protocols begin to arise. Disruptions in service continuity due to handoffs is an unacceptable outcome. Thus, it is essential to address these issues in mobile environments. Although technology is constantly evolving, TCP has remained mostly the same for more than 20 years [1]. In fact, 95% of all internet traffic is TCP [2]. These reasons have motivated the introduction of Multi-Path TCP (MPTCP) as a possible solution to the mobile scenario.

This work sheds light on overlooked handoff issues and proposes a Cross-Layer
Assisted MPTCP (CLA-MPTCP) to alleviate resulting performance issues and glitches. Leveraging cross-layer information, where layers exchange data, improves MPTCP responsiveness, throughput and latency during fluctuating network conditions. Contributions of this work are:

- Performance evaluation and analysis of transport protocols during handoffs in HetNets.
- Establishing that reducing congestion windows at the time of handoff and utilizing cross-layer assistance is essential for ensuring service continuity.
- Design of handoff-aware cross-layer assisted MPTCP coupled congestion control algorithm.
- To our knowledge, this work is the first to investigate handoff connection disruptions and glitches with MPTCP, and to improve service continuity during handoffs through proactive cross-layer assisted SNR and BDP-based congestion window adjustments.

The rest of this paper is organized as follows. In Section 3.2 an experiment is conducted and findings are presented to analyze handoff issues in currently deployed wireless networks. MPTCP as a potential solution to handoff issues is discussed in Section 3.3. CLA-MPTCP is presented and evaluated in Sections 3.4 and 3.5. Finally the article is concluded in Section 3.6.
3.2 Handoff Induced Issues in Current HetNets

Some work has been done in investigating transport layer issues during handovers such as those in [3], [4], [5], [6]. For instance, traditional TCP has many handoff issues in terms of mobile data transfers, connection glitches and service continuity [7]. Sinky and Hamdaoui [7] examine and propose solutions to these issues using the New Reno variant of TCP which is the most widely used variant today. The challenge is to seamlessly maintain a mobile user’s connection with little to no perceivable interruption in the event of a handoff. These issues are among many that have motivated the introduction of MPTCP. Since MPTCP utilizes multiple TCP subflows it is essential to understand basic TCP handoff induced issues.

Different TCP problems are amplified depending on the type of handoff that occurs [3], [4], [5], [6], [7]:

3.2.0.1 High-delay network to low-delay network

- **Packet re-ordering:** This issue occurs when the source node switches to a considerably faster link. Packets sent on the new link overtake packets that have been sent on the old link which causes out of order packets to arrive at the destination, resulting in duplicate acknowledgments (dupacks) to be sent. TCP would misinterpret 3 dupacks as being caused by congestion or loss and re-transmit packets incorrectly assumed to be lost when in reality this was caused by a handoff.
• **Inflated re-transmission timeout (RTO):** Naturally, a high-delay link will have high RTT and RTO values. Since the RTO is updated once every RTT, when a handoff occurs, the RTO will converge slowly to the new low RTO value. This is an issue because invoking RTO recovery to recover lost packets will take longer. In other words, in the event of lost packets on a low-delay link, TCP would not recover quickly due to the high RTO value of the high-delay link. Thus, TCP must be able to recover lost packets faster using a smaller RTO representing the new low-delay link.

3.2.0.2 Low-delay network to high-delay network

• **Spurious or false re-transmission timeout (RTO):** This issue arises when a handoff occurs to a slower link. The low-delay link before the handoff has a low RTO value. Packet acknowledgments before the handoff take the high-delay link after the handoff. This causes TCP to falsely or spuriously timeout assuming packets have been lost due to a low RTO value from the low-delay link.

• **Link overshoot:** In high-delay networks TCP requires several RTTs to probe available bandwidth. Specifically, after a handoff to a high-delay link the sender may inject more data onto the link than can be handled resulting in dropped packets. In addition, when TCP is in the slow start phase the congestion window grows exponentially. If a handoff occurs during the slow start phase a congestion window increase may result in packet losses and
timeouts. This is referred to as a slow start overshoot. TCP may interpret this incorrectly and enter its congestion avoidance phase with a very small congestion window resulting in suboptimal TCP performance.

Transport layer protocols are neutral to the varying technologies of HetNets; rather, they react to the qualities of service and data rates available. Current deployed networks struggle to maintain service continuity, consistency and performance in mobile scenarios resulting in a frustrating user experience which is illustrated in 3.2.1.

3.2.1 Transport Layer Analysis

A preliminary experiment was designed to force a mobile device into multiple handoffs. The HetNet in the EECS department at Oregon State University consists of 41 access points using both the 2.4 and 5 GHz bands and 802.11 a, b, g and n access technologies. The experiment was limited to the third floor, which comprised of 12 access points. A python script logged network measurements while the mobile device downloaded a large file and followed a path involving multiple access points as shown in Figure 3.1a. Figure 3.1 uses one of the experimental runs to illustrate these findings.

Access points throttle down transmit rates while mobile devices increase their power in an effort to maintain the connection resulting in performance degradation. Mobile terminals tend to cling to their current access point unless a handoff is necessary skipping intermediate access points across their path. This causes
Figure 3.1: Transport layer performance measurements related to multiple handoffs.
performance issues when multiple handoffs take place as shown by the erratic signal-to-noise ratio (SNR), throughput and RTT behavior around the points of handoff in Figure 3.1. Each vertical line indicates a channel switch or handoff logged by the mobile terminal as it moves across the path.

A mobile user can undergo multiple handoffs during an active connection’s lifetime. This involves experiencing the quality degradation of a depleting data connection until a handoff is complete. That is, SNR and throughput decrease while RTTs increase as a handoff is approached and initiated. This is followed by a period of satisfactory connection quality until the connection begins to deplete again usually ending when a mobile user becomes stationary. This cycle greatly affects performance resulting in throughput gaps and service continuity issues for mobile users.

Transport layer protocols must be able to differentiate between delay caused by actual packet loss and congestion on a link and delay simply caused by handoffs. A combination of metrics must be considered for an accurate representation of network conditions. Section 3.3 briefly introduces MPTCP and some of its drawbacks.

3.3 Multi-Path TCP (MPTCP)

Despite being a relatively new protocol, MPTCP is becoming increasingly popular. With the release of iOS 7 in 2013, Apple became the first to implement MPTCP commercially. The novelty behind MPTCP is its ability to decouple TCP and
IP to simultaneously use multiple TCP subflows and interfaces transparent to the application [1].

Among the main modes of operation available to MPTCP are fullmptcp, backup and singlepath [8], [9]. In fullmptcp mode the connection’s data is striped among all subflows as space in the subflow windows becomes available where most of the data is sent on the least congested subflow (Linux MPTCP implements a path manager that schedules data on subflows with the lowest RTT) [9], [10], [11]. In backup mode slave subflows that have joined the current MPTCP connection are only used when the master subflow, used for initiating the MTPCP connection, fails [8], [12]. In singlepath mode a new subflow is established only when the interface goes down.

3.3.1 MPTCP Architecture

A simplified overview of MPTCP can be seen in Figure 3.2. MPTCP’s inherent architecture provides a potential solution to the aforementioned handoff issues. For instance, an application on a smartphone may use a single MPTCP connection to utilize both Wi-Fi and cellular interfaces to communicate with a server even if endpoints were to change, whereas traditional TCP would break.

MPTCP subflows may appear or disappear at any time during an active connection. Similar to TCP, each subflow is initiated using a three-way handshake and operates independently of others with their own congestion states (i.e. cwnd) [12]. For reliable, in-order data delivery MPTCP uses a data sequence number
(DSN) to number all data sent over a MPTCP connection and a per subflow sequence number (SSN) mapped to the DSN. This allows for the same data (DSN) to be re-transmitted on different subflows in the event of packet loss or failure. The mapping, once declared, is fixed and carried with the SSN sent. DSN are acknowledged using a data connection level acknowledgment.

MPTCP aims to simultaneously pool available resources, appearing as a single resource to the end user application, whilst greatly improving user experience. This is realized through three MPTCP goals listed in [13]:

- **Goal 1** (Improve throughput): Perform at least as well as single-path TCP would on its best subflow.

- **Goal 2** (Do no harm): Do not use more capacity than if it was single-path TCP on its best subflow.

- **Goal 3** (Balance congestion): MPTCP should move as much traffic as possible off its most congested paths.

### 3.3.2 MPTCP Congestion Control

Simply running standard TCP congestion control on each subflow gives MPTCP an unfair share of capacity when its subflows share a bottleneck [13]. By default, MPTCP implements a coupled congestion control algorithm that links subflow increase functions. For each ACK on subflow $i$, congestion window $w_i$ is increased by $\min(\alpha \frac{w_{\text{total}}}{w_i}, \frac{1}{w_i})$. For each loss $w_i = \frac{w_{\text{old}}}{2}$. The $\alpha$ parameter controls the aggressiveness
of a MPTCP connection and is chosen so that the MPTCP aggregate flow is equal to the achievable throughput of a single-path TCP flow on the best path [13]:

\[ \alpha = \frac{w_{\text{total}} \times \max \left( \frac{w_i}{RTT_i^2} \right)}{\left( \sum w_i / RTT_i \right)^2} \]

### 3.3.3 Drawbacks of MPTCP Congestion Control

Handoff issues presented in Section 3.2 still apply to the TCP subflows of MPTCP. For example, MPTCP reordering occurs both at the subflow and data levels which can cause added complexity with duplicate ACKs and spurious RTOs. Common MPTCP drawbacks that may arise from handoffs are listed below:

- **Responsiveness**: MPTCP relies on \( \alpha \) to adapt to network conditions. However, to reduce computational costs, \( \alpha \) is only updated when there is a packet
drop or once per RTT [14]. In addition, sender RTT estimations can be inaccurate due to TCP’s delayed ACK mechanism and smoothed RTT values (SRTT) especially when a network is over-buffered. This results in slow responsiveness to changes in subflow congestion windows resulting in an underestimation of $\alpha$ affecting MPTCP’s increase rate. Thus, a depleting connection due to handoff may be misinterpreted by MPTCP as being highly congested. The time needed for RTT updates to characterize a congested link is detrimental for a highly mobile user. With the advantage of cross-layer assistance this issue can be minimized by taking into consideration physical network conditions such as signal strength and user mobility that otherwise are not available to MPTCP. By reducing the dependence on RTT and packet loss as a means to adjust MPTCP aggressiveness, sensitivity to a depleting connection due to handoffs is increased allowing for quicker adjustments to be made and more seamless service continuity.

- **Fullmesh gaps**: In fullmesh mode data that was originally striped onto a particular subflow is delayed or lost at the time of handoff and would need to be re-mapped onto other subflows. The worst case scenario is a subflow’s entire congestion window is lost resulting in a gap in the received data sequence numbers. These data sequence numbers are then re-mapped to other available subflows causing glitches and affecting service continuity at the point of handoff. Delay is increased even more if RTOs are high forcing subflows to be momentarily inactive. Incorporating cross-layer assistance will allow for data originally mapped onto the conflicting subflow to be re-
mapped earlier onto other available subflows during the scheduling process avoiding increased delays and large gaps in the received sequence numbers.

- **Backup delay**: In backup mode packets are transmitted on different sub-flows only if the master subflow fails. This is not ideal for handoff scenarios as it negatively affects service continuity by waiting for a connection to fail before using another [8]. The ability to detect a failing subflow through cross-layer assistance can allow MPTCP to utilize subsequent subflows prior to the master subflow failing.

Packet loss and RTT alone are not enough for a MPTCP connection to quickly adapt to handoffs. Thus, balancing between network congestion and impending handoff as a congestion control mechanism is necessary. A solution is required that incorporates not only real-time network monitoring but also cross-layer assistance in MPTCP. In the following section a Cross-Layer Assisted MPTCP (CLA-MPTCP) is proposed to alleviate handoff induced glitches and disruptions in service continuity.

### 3.4 Cross-Layer Assisted Coupled Congestion Control

MPTCP requires the ability to differentiate between fluctuations in network conditions caused by congestion and those caused by mobility and network handoffs. Figure 3.3 shows subflow SNR of a typical MPTCP connection as a mobile host moves across a path while communicating with a server. This results in a scenario
where stable conditions are experienced on the cellular interface and erratic behavior on the Wi-Fi interface where gaps in SNR, throughput and service continuity are seen. As mentioned before, relying on RTT alone as a measure of link quality does not allow MPTCP to adapt quickly enough especially when RTT and RTO values are high.

In order to minimize connection glitches and maintain adequate service continuity throughout the handoff process the bandwidth delay product (BDP) must be considered [7]. The BDP refers to the maximum amount of data that can be sent on a network link at any given time. In other words, it represents the amount of data that is sent before the first ACK is received [15]. The BDP is represented by the product of a link’s capacity and its RTT. By definition then the path BDP, $BDP_{path}$, is the product of the bottleneck throughput of the forward and backward paths, $R_{min}$, and the path round-trip time, $RTT_{path}$\(^1\); i.e., $BDP_{path} = R_{min} \times RTT_{path}$. When propagation and queuing delays are neglected, $RTT$ consists of propagation, transmission, queuing and processing delays only; i.e., end-to-end packet delay.
we can write

$$BDP_{path} = R_{min} \left( \sum_{i=1}^{n} \frac{S}{R_i} + \sum_{i=1}^{m} \frac{S'}{R_i'} \right)$$

(3.1)

where $n$ and $R_i$ are the number of hops and throughput on the forward path, $m$ and $R_i'$ are the number of hops and throughput on the backward path, and $S$ and $S'$ are the packet sizes in the forward and backward paths respectively. As done in [15], since $\sum_{i=1}^{n} \frac{S}{R_i} \leq \sum_{i=1}^{n} \frac{S}{R_{min}}$ and $\sum_{i=1}^{m} \frac{S'}{R_i'} \leq \sum_{i=1}^{m} \frac{S}{R_{min}}$, the path BDP can be loosely upper bounded by

$$BDP_{path} \leq R_{min} \left( \sum_{i=1}^{n} \frac{S}{R_{min}} + \sum_{i=1}^{m} \frac{S}{R_{min}} \right) = nS + mS = S \times N$$

(3.2)

where $N = n + m$. Note that having the path BDP exceed $S \times N$ results in packets being queued at the bottleneck [15]. In what follows, let $BDP_{UB} = S \times N$.

At the time of handoff, where service continuity is crucial for a user’s quality of experience (QoE), sending no more than the $BDP_{UB}$ helps reduce queuing times, RTT values and jitter while maintaining adequate throughput [7], [16]. In contrast to our previous work, which considered traditional single-path TCP, we shift our focus towards MPTCP and improving service continuity through proactive BDP and SNR-based subflow congestion window adjustments as a solution to the mobile handoff problem.
3.4.1 Design

Modifications to MPTCP are proposed to alleviate the issues presented in Section 3.2. An additional MPTCP design goal is introduced:

- **Design goal** (Cross-layer assistance) A multi-path flow should consider both a subflow’s signal strength and BDP to anticipate and alleviate handoff induced network conditions to maintain and improve service continuity.

In designing CLA-MPTCP three assumptions are made:

- A mobile device is equipped with two interfaces (i.e. cellular and Wi-Fi) where the master subflow is initiated on the Wi-Fi interface.
- Multiple handoffs are experienced on the Wi-Fi interface across a path.
- Both endpoints of a CLA-MPTCP connection exchange, through signaling, cross-layer information such as current and candidate access point signal strengths and handoff notifications.

CLA-MPTCP brings three modifications to MPTCP:

- Utilization of a signal strength-based subflow throughput model to prepare for handoffs while maintaining a connection-level throughput threshold \( R_0 \).
- Subflow congestion windows are adjusted to \( BDP_{UB} \) and \( RTT \) and \( RTO \) estimations are reset on handoff notifications.
- Couples interface signal strength and subflow throughput with MPTCP’s congestion control algorithm to preemptively adjust the aggressiveness of individual subflows in anticipation of handoffs.

All subflows on a particular interface must go through their point of access network as shown in Figure 3.4. That is, a single interface cannot be connected to multiple cellular towers or multiple Wi-Fi access points. In the event of a handoff all subflows on the current interface must disconnect and reconnect on the next access network. For instance, cellular GSM networks, which make up more than 80 percent of all global mobile connections, implement hard handoffs where a connection must be broken until it is reestablished [17]. Conversely, handoffs between access points within the same Wi-Fi network can maintain their connection but still suffer from the consequences of a depleting connection. This requires more complex and adaptive subflow policies in order to maintain service continuity.
3.4.2 System Model

Compared to Wi-Fi, cellular connections experience higher latency and lower throughput while providing stability and less packet loss. On the other hand, the time it takes for a roaming client’s Wi-Fi interface to scan and connect to available access points depends on many factors and can be detrimental to service continuity. IEEE standards such as 802.11k and r aim to streamline this process to help reduce roaming times. The 802.11k amendment reduces scan times by providing roaming clients with a neighbor report listing candidate access points. In addition, 802.11r expedites the authentication process by allowing a client to pre-authenticate with an access point potentially reducing authentication time from seconds to milliseconds.

Commercial implementations, such as those in Apple and Google mobile devices, incorporate a signal strength trigger threshold\(^2\) that, once reached, is used to initiate a scan for candidate access points provided by 802.11k. Typically, a candidate access point with a signal strength differential of 12 dBm is selected triggering the handoff and pre-authentication using 802.11r. A signal strength differential is necessary as to avoid a client from ping ponging between candidate access points. The proposed system model considers these aforementioned concepts in designing CLA-MPTCP.

The model parameters defined in Table 3.1 are integrated within CLA-MPTCP to adaptively maintain the connection threshold \(R_0\) by conducting subflow signal

\(^2\)The trigger threshold is the minimum signal strength a client requires to maintain a connection. For example, Apple devices use an RSS trigger threshold of -70 dBm.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>Capacity of Wi-Fi interface</td>
</tr>
<tr>
<td>$C_2$</td>
<td>Capacity of cellular interface</td>
</tr>
<tr>
<td>$R_1^1$</td>
<td>Current Wi-Fi throughput on $AP_1$</td>
</tr>
<tr>
<td>$R_2^2$</td>
<td>Current Wi-Fi throughput of candidate access point ($AP_2$)</td>
</tr>
<tr>
<td>$R_{th}^1$</td>
<td>Wi-Fi throughput at time of handoff</td>
</tr>
<tr>
<td>$R_2$</td>
<td>Current cellular throughput</td>
</tr>
<tr>
<td>$R_{ph}^2$</td>
<td>Cellular throughput when pre-handoff is initiated</td>
</tr>
<tr>
<td>$R_{tot}$</td>
<td>Total connection throughput = $\sum R_i$</td>
</tr>
<tr>
<td>$R_0$</td>
<td>Connection throughput threshold</td>
</tr>
<tr>
<td>$\delta$</td>
<td>$R_0 - R_1^1 - R_2$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>Current signal strength of $AP_1$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>Current signal strength of $AP_2$</td>
</tr>
<tr>
<td>$\Delta_0$</td>
<td>Scan trigger threshold in dBm</td>
</tr>
<tr>
<td>$\Delta_1$</td>
<td>Pre-handoff trigger threshold in dBm</td>
</tr>
<tr>
<td>$\Delta_2$</td>
<td>Handoff trigger threshold in dBm</td>
</tr>
</tbody>
</table>

Table 3.1: CLA-MPTCP system parameters
strength-based throughput adjustments and provide a more seamless handoff transition. Using the measurement findings from Section 3.2, CLA-MPTCP categorizes a mobile user into three distinct stages: stationary, pre-handoff and handoff. The design of these stages are discussed next.

3.4.2.1 Stationary

During the stationary stage CLA-MPTCP subflows are assumed to maintain stable throughput, that is, achieving their respective interface capacities $C_i$. In order to maintain connection threshold $R_0$, the CLA-MPTCP network module monitors and exchanges subflow goodput $R_i$ and current and candidate AP signal strengths $S_i$ between source and destination. Data is offloaded onto the cellular interface if $R_{tot} < R_0$ and $S_1 > S_2 + \Delta_1$ where cellular throughput $R_2 = \max (0, \min (\delta, C_2))$.

The stationary process is illustrated in Figure 3.5. This differs from the pre-handoff stage where subflow rates are proactively adjusted based on their current rates and signal strengths.

3.4.2.2 Pre-Handoff

During this phase proactive preparations are made in anticipation of a handoff. This process assumes a depleting connection on the Wi-Fi interface where throughput decreases proportional to the signal strength. All internet routers contain buffers to hold packets in times of congestion [18] and due to immense bufferbloat
in real networks [14], the amount of actual in-flight packets is much higher than the real path BDP. Thus, in order to reduce a subflow’s throughput, in response to mobility and handoffs, the congestion window needs to be proactively reduced below the path BDP. The intuition behind this is to gradually reduce buffer sizes, queuing delays and round trip times in order to provide a more seamless transition.

CLA-MPTCP’s subflow throughput, $R_i$, is modeled as a function of signal strength, $S_i$, using a system of linear equations. The design is formulated based on the desired behavior illustrated in Figure 3.6 where the x and y axes represent Wi-Fi signal strength over time and interface capacities respectively. When pre-handoff is initiated, that is, when conditions $R_{tot} < R_0$ and $S_1 < S_2 + \Delta_1$ are met, the Wi-Fi subflow is assumed to have maintained a stable throughput of $C_1$.
up to this point. $R^h_1$ is a tunable parameter representing the rate reached at the time of handoff. Note that prior to a handoff a Wi-Fi subflow still achieves usable throughput, where $R^h_1$ may not necessarily be zero. During pre-handoff, the desired Wi-Fi subflow rate, $R^1_1$, can be derived using the following linear system:

$$R^1_1(S_1) = \begin{cases} 
C_1 = a \times (S_2 + \Delta_1) + b \\
R^h_1 = a \times (S_2 - \Delta_2) + b 
\end{cases}$$

The adjusted pre-handoff rate for the Wi-Fi interface is then,

$$R^1_1 = \frac{C_1 - R^h_1}{\Delta_1 + \Delta_2} \times (S_1 - S_2 - \Delta_1) + C_1$$

In addition, upon entering pre-handoff, cellular throughput, $R^p_2$, is set to the current $R_2$ from the stationary phase. When a handoff is initiated, the amount of
data offloaded onto the cellular interface is maximized to its respective capacity $C_2$ in order to reduce service continuity issues. As before, this gives the following linear system:

$$R_2(S_2) = \begin{cases} 
R_p^2 = a \times (S_1 - \Delta_1) + b \\
C_2 = a \times (S_1 + \Delta_2) + b 
\end{cases}$$

The adjusted pre-handoff rate for the cellular interface is then,

$$R_2 = \frac{C_2 - R_p^2}{\Delta_1 + \Delta_2} \times (S_2 - S_1 - \Delta_2) + C_2$$

This phase provides CLA-MPTCP with the desired Wi-Fi and cellular subflow adjustments needed to prepare for and anticipate handoffs. CLA-MPTCP subflow congestion windows are then proactively adjusted using $BDP_{path} = R_{new} \times RTT_{path}$ where $R_{new}$ is set to the newly derived throughputs, $R_1$ and $R_2$. This allows for $R_0$ to be maintained as long as possible until a handoff is initiated.

A large congestion window leading up to a handoff can be detrimental to service continuity [7]. Maintaining a large congestion window increases the probability of contention, RTT, queuing and packet loss which can degrade MPTCP performance drastically. By setting the congestion window to the adjusted path BDP, that is the maximum number of bytes that can be in transit across the path, guarantees the transmission of the maximum amount of bytes while avoiding additional packet queuing at intermediate links at the time of handoff. Figure 3.7 summarizes the pre-handoff process.
Figure 3.7: CLA-MPTCP pre-handoff stage flow
3.4.2.3 Handoff

When a handoff is initiated the Wi-Fi interface must disconnect and re-associate with a neighboring access point. Naturally, this causes the current throughput to momentarily be zero until a re-association completes. However, prior to the handoff the connection may still be achieving usable throughput. Traditionally, the scan and handoff trigger thresholds, $\Delta_0$ and $\Delta_2$, take precedence in deciding when to handoff. The handoff phase adds two checks to the traditional method. If $\Delta_0$ has not been reached a request for an AP list is made if $R_{tot} < R_0$. Assuming the current and candidate access points share similar characteristics and capacities, the current rate of the candidate access point $R^2_1$ can similarly be modeled in terms of signal strength:

$$R^2_1(S_2) = \begin{cases} 
0 = a \times (S_1 - \Delta_1) + b \\
C_1 = a \times (S_1 + \Delta_2) + b 
\end{cases}$$

Yielding,

$$R^2_1 = \frac{C_1 - R^h_1}{\Delta_1 + \Delta_2} \times (S_2 - S_1 - \Delta_2) + C_1$$

$R^2_1$ is continually monitored and compared to the current rate $R_1^1$. If $R_1^2 > R_1^1$ or $S_2 > S_1 + \Delta_2$, whichever is true first, an association with the candidate AP is initiated.

When a CLA-MPTCP sender receives a subflow handoff notification it resets
Figure 3.8: CLA-MPTCP handoff stage flow
RTT and RTO estimations. The intuition behind this is that if the RTO is set too high MPTCP may be slow to recover lost packets. If set too low, spurious RTOs cause MPTCP to assume packets are lost when in reality they may simply be on their way. Analyses established that by resetting these values MPTCP converges quicker to the new network’s RTT and RTO estimations. Depending on the subflow TCP variant, duplicate acknowledgments are temporarily disabled to avoid false dupacks caused by packet reordering. Finally, the congestion window of the subflow is set to the respective path BDP to avoid added queuing at the bottleneck as well as link overshoots. Using the path BDP ensures throughput is not compromised as shown in Section 3.5.

In Section 3.5 service continuity and performance measurements (i.e. aggregate throughput, RTT, jitter, re-transmissions and duplicate ACKs) are evaluated.

3.5 CLA-MPTCP Evaluation

To evaluate CLA-MPTCP the network topology in Figure 3.11a is built using NS3-DCE. DCE is a framework that allows for existing Linux *kernel space* network protocols to be used within NS3 simulations [19]. In addition, an access point decision algorithm is implemented in NS3 that uses an SNR-based scan trigger threshold, $\Delta_0$, of 20 dBm. Once reached, the mobile client scans for candidate access points and initiates an association if $S_2 > S_1 + \Delta_2$ where $\Delta_2 = 12$ dBm. CLA-MPTCP modifications were implemented in version 0.87 of the MPTCP Linux Kernel implementation and installed on NS3 nodes using DCE. Each CLA-MPTCP user
Figure 3.9: CLA-MPTCP throughput performance using different $R_0$ values compared to Uncoupled Reno, Coupled, and Olia using traditional handoff scheme with $\Delta_0 = 20$ and $\Delta_2 = 12$ dbm.
Figure 3.10: Average RTT, re-transmissions, duplicate ACKs and jitter measurements

Figure 3.11: (a) NS3-DCE Network Topology used for simulations. (b) Throughput loss at handoff as a function of $R_0$. 
is equipped with two interfaces, i.e. Wi-Fi and cellular. Similar to the analysis scenario in Section 3.2.1, the mobile client moves across a path and undergoes two Wi-Fi handoffs while remaining connected to the LTE network. Data is sent from the mobile source to the destination at 30 Mbps where the capacity and delay of the Wi-Fi and LTE networks are set to 25 and 18 Mbps, 5 and 20 milliseconds respectively. In addition, $R^h_1$ and $\Delta_1$ are set to 5 Mbps and 25 dBm. The selection of the emulation parameters shown in Table 3.2 were chosen based on our previous measurement study of currently deployed networks conducted in Section 3.2 and Figure 3.1b as well as manufacturer suggestions. For instance, parameters $\Delta_0$ and $\Delta_2$ were selected based on manufacturer recommendations of what is considered a suitable scan and handoff trigger threshold [20], [21]. Conversely, $\Delta_1$ is an implementation specific decision chosen to balance the detection sensitivity between the stationary and pre-handoff phases. Thus, having $\Delta_1 > \Delta_0$ allows for CLA-MPTCP to enter the pre-handoff phase prior to scanning for candidate access points. Multiple simulations are run using different $R_0$ (connection throughput threshold) values ranging from 10 Mbps to $C_1$. Finally, network information such as current and candidate access point signal strength, subflow goodput and handoff notifications are exchanged between source and destination. During the CLA-MPTCP connection, the NS3-DCE mobile devices periodically log access point signal strengths in order to calculate the equations associated with the current phase. Algorithm 3 is applied in the stationary stage whereas algorithms 4 and 5 are applied in the pre-handoff and handoff stages. Once calculated the updated rates are exchanged with the CLA-MTPCP congestion control module
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>25 Mbps</td>
</tr>
<tr>
<td>$C_2$</td>
<td>18 Mbps</td>
</tr>
<tr>
<td>$R_{\frac{h}{i}}$</td>
<td>5 Mbps</td>
</tr>
<tr>
<td>$R_0$</td>
<td>[10-24] Mbps</td>
</tr>
<tr>
<td>$\Delta_0$</td>
<td>20 dBm</td>
</tr>
<tr>
<td>$\Delta_1$</td>
<td>25 dBm</td>
</tr>
<tr>
<td>$\Delta_2$</td>
<td>12 dBm</td>
</tr>
</tbody>
</table>

Table 3.2: CLA-MPTCP emulation parameters

for proactive congestion window adjustments. Information exchange can be implemented either using MPTCP’s header option or through the use of Linux kernel external variables. For simplicity, the emulation environment is setup to utilize MPTCP specific external variables as well as toggling subflow binary variables for handoff notifications thus adding negligible overhead. Note MPTCP’s second goal limits the aggregate throughput to no more than what is achievable by a single TCP flow on its best subflow (Wi-Fi).

The achieved aggregate goodput of CLA-MPTCP is compared with Uncoupled, Coupled and Olia (Opportunistic Linked Increases Algorithm) versions of MPTCP. Uncoupled MPTCP runs traditional TCP Reno on its individual subflows whereas coupled MPTCP implements the coupled congestion control algorithm which links the increase functions of MPTCP subflows introduced in Section 3.3.2. Olia increases good quality subflows with small windows faster while subflows with maximum windows increase slower. In other words, Olia re-forwards traffic from fully used paths to paths that have free capacity [22]. CLA-MPTCP integrates station-
Algorithm 3 CLA-MPTCP Stationary Stage

1: **Input:** $R_0, S_i, C_i, \Delta_1, \Delta_2, \delta, R_{tot}$
2: **while** $S_1 > S_2 + \Delta_1$ and $R_{tot} < R_0$ **do**
3:     **for** each subflow $i$ **do**
4:         **if** path index $==$ cellular interface **then**
5:             $R_{tot}+ = R_i$
6:             let $\delta = R_0 - R_1^l - R_2$
7:             let $R_2 = \max (0, \min (\delta, C_2))$
8:             let $R_{new} = R_2$
9:         **end if**
10:         **if** path index $==$ Wi-Fi interface **then**
11:             $R_{tot}+ = R_i$
12:             let $R_{new} = R_1^l$
13:         **end if**
14:     calculate
15:         $BDP_{path} = \frac{R_{new} \times RTT_i}{MSS_i}[pkts]$
16:     **set** $cwnd_i = BDP_{path}$
17: **end for**
18: **end while**
19: re-calculate $\alpha$ using adjusted cwnds
Algorithm 4 CLA-MPTCP Pre-Handoff Stage

1: **Input:** $R_0, S_1, C_i, R_{1}^h, R_{2}^p, \Delta_1, \Delta_2, R_{tot}$

2: From 802.11k AP report:
   - let $S_1 = \text{current } AP_1 \text{ signal strength}$
   - let $S_2 = \text{candidate } AP_2 \text{ signal strength}$

3: **while** $R_{tot} < R_0$ and $S_1 < S_2 + \Delta_1$ **do**

4:     **for** each subflow $i$ **do**

5:         **if** path index == cellular interface **then**

6:             $R_{tot} + = R_i$

7:             calculate

8:                 $$R_2 = \frac{C_2 - R_{2}^p}{\Delta_1 + \Delta_2} \times (S_2 - S_1 - \Delta_2) + C_2$$

9:         **end if**

10:     **if** path index == Wi-Fi interface **then**

11:         $R_{tot} + = R_i$

12:         calculate

13:             $$R_1^l = \frac{C_1 - R_{1}^h}{\Delta_1 + \Delta_2} \times (S_1 - S_2 - \Delta_1) + C_1$$

14:         **end if**

15:     calculate

16:         $$BDP_{path} = \frac{R_{new} \times RTT_i}{MSS_i} \text{[pkts]}$$

17:         set $cwnd_i = BDP_{path}$

18:     **end for**

19: **end while**

20: re-calculate $\alpha$ using adjusted cwnds
Algorithm 5 CLA-MPTCP Handoff Stage

1: **Input:** $R_0, R^h_1, R^1_1, S_i, C_i, \Delta_1, \Delta_2, R_{tot}$
2: **while** $S_1 < \Delta_0$ or $R_{tot} < R_0$ **do**
3: request 802.11k AP report
4: let $S_2$ = candidate $AP_2$ signal strength
5: calculate
   \[ R^2_1 = \frac{C_1 - R^h_1}{\Delta_1 + \Delta_2} \times (S_2 - S_1 - \Delta_2) + C_1 \]
6: **if** $S_2 > S_1 + \Delta_2$ or $R^2_1 > R^1_1$ **then**
7: initiate 802.11r authentication
8: associate with $AP_2$
9: notify CLA-MPTCP sender of handoff
10: **end if**
11: **end while**

... 

12: **if** Handoff notification on subflow $i$ **then**
13: reset $RTT_i$ and $RTO_i$ estimations
14: temporarily disable dupacks
15: obtain path $BDP_{UB}$
16: set $cwnd_i = BDP_{UB}$
17: **end if**
ary, pre-handoff and handoff stages whereas existing implementations incorporate the traditional handoff method with $\Delta_0 = 20$ and $\Delta_2 = 12$ dBm.

Figure 3.9 shows LTE, Wi-Fi and aggregate throughput for different $R_0$ values. With different $R_0$ values the amount of data offloaded onto the LTE subflow as well as aggregate throughput can be tuned. That is, as $R_0$ increases more data is offloaded near handoffs resulting in a seamless transition for the end user. Results show CLA-MPTCP quickly adapts to network conditions and proactively offloads data from the Wi-Fi subflow to the LTE subflow. CLA-MPTCP’s ability to maintain $R_0$ as well as preemptively tune subflow congestion windows to the adjusted path BDP ensures improved throughput and softens the transition. Essentially, $R_0$ is a parameter that fills the gaps in throughput caused by handoffs to improve a mobile device’s service continuity.

Figure 3.11b shows the amount of throughput loss at the point of handoff as a function of $R_0$ using different $\Delta_2$ values ranging from 1 to 15 dBm. CLA-MPTCP will try to maintain at least $R_0$ throughout the handoff process. If, at the time handoff, the aggregate throughput is greater than $R_0$ a negative throughput loss is experienced. For example, when $R_0 = 18$, CLA-MPTCP achieves an average aggregate throughput of 18.8 Mbps which results in a negative throughput loss or, in other words, a throughput improvement. Conversely, when $R_0 = 24$, the aggregate throughput cannot exceed the maximum achievable throughput at the time of handoff, $R_1^h + C_2$, which is less than $R_0$ resulting in a positive throughput loss. CLA-MPTCP’s integration of a connection level threshold results in less throughput loss near handoffs where existing implementations lose throughput
drastically as \( R_0 \) increases due to the lack of cross-layer assistance and handoff awareness.

Since current Linux MPTCP congestion control algorithms, uncoupled, coupled and Olia, rely on a path’s RTT and packet losses when striping an input stream among the available subflows they do not react fast enough to sudden handoffs across a mobile user’s path. The RTT of a subflow at the point of handoff may still be less than the RTT on other subflows causing MPTCP to still prefer and cling to the depleting subflow. In other words, the path scheduler within Linux MPTCP schedules data on the subflow with the lowest RTT value. For instance, although uncoupled Reno utilizes two separate subflows it still schedules most of the data on path with the lowest RTT value which happens to be the Wi-Fi network in the emulation environment. This results in similar behavior in their observed performances in Figures 3.9 and 3.10.

Figure 3.10 shows RTT, re-transmission, duplicate acknowledgment and jitter behavior of CLA-MPTCP. A near 50% and 75% improvement in RTT and jitter can be seen around the points of handoffs. In addition, CLA-MPTCP sees improvements in the number of packet re-transmissions and duplicate acknowledgments experienced. A sharp dip in packet re-transmissions can be seen at the time of handoff as well as stable and low occurrences of duplicate acknowledgments. Conversely, a sharp increase in packet re-transmissions occurs following a handoff due to the mobile device re-associating with a candidate AP when \( R_i^2 > R_i^1 \). This condition is met to prevent waiting for the trigger threshold, \( \Delta_0 \), and candidate AP differential, \( \Delta_2 \), to be reached. The re-association and utilization of the Wi-Fi
network occurs earlier near the edge of the candidate AP’s range which explains the temporary increase in packet re-transmissions following handoffs. The proactive congestion window adjustments made by CLA-MTPCP results in less packets being injected into the network near handoffs and a less strenuous transition.

3.6 Conclusion

In this paper, issues dealing with handoffs in HetNets were discussed. Specifically weaknesses of transport layer protocols in mobile scenarios where multiple handoffs are imminent. A series of real world experiments in current deployed networks were conducted exploiting traditional TCP in mobile scenarios. MPTCP as a potential solution to the handoff issue is discussed. Possible drawbacks of MPTCP in increasingly mobile scenarios are presented. An additional MPTCP design goal is suggested emphasizing the need for cross-layer assistance. Finally, Cross-Layer Assisted MPTCP (CLA-MPTCP) is designed and evaluated. Results show the benefits of cross-layer assistance and integration of a connection level threshold and subflow signal strength as MPTCP outperforms uncoupled, coupled andolia congestion control algorithms in scenarios where multiple handoffs take place.
Chapter 4: Large Content-Centric Urban Communication Networks

Proof of Concept Manuscript

Cloudlet-Aware Mobile Content Delivery in Wireless Urban Communication Networks

Hassan Sinky and Bechir Hamdaoui
School of Electrical Engineering and Computer Science, Oregon State University,
Corvallis, Oregon 97331, Email: sinkyh,hamdaoub@oregonstate.edu

ABSTRACT

LinkNYC is a first-of-its-kind urban communications network aiming to replace all payphones in the five boroughs of New York City with kiosk-like structures providing free super fast gigabit Wi-Fi to everyone. This work proposes and investigates the applicability of shifting LinkNYC from a traditional IP network to a content-centric network by upgrading all or a subset of their kiosks with standalone cloudlets, which cache content as it disseminates throughout the network, and content delivery cloudlets which are geographically distributed throughout the boroughs and store popular internet content. With this shift content is brought much
closer to the end user than traditional methods which is essential in highly mobile environments. Analysis shows that adopting multiple content delivery cloudlets dramatically improves overall network performance and stability. Finally, given a cloudlet-aware path a mobile content delivery scheme is designed to offset service continuity issues that are amplified when a mobile user encounters multiple cloudlets with intermittent connectivity. Results show an overall improvement in a mobile user’s throughput, response time and cache hit percentage.

4.1 Introduction

The proliferation of internet devices has resulted in efforts to integrate various wireless access technologies for improved performance, increased services and interconnectivity of end users. The recent growth in data demand has prompted researchers to come up with new wireless techniques (e.g., MIMO [1], cooperative communication [2], femtocells [3], etc.) and develop new technologies (e.g., cognitive radio [4], LTE [5], etc.) to be able to meet this high demand. This integration allows for large geographic locations to be serviced providing millions of end users with continuous connectivity and optimal quality of experience (QoE). However, the world has seen unprecedented urban population growth over the years. In fact, the number of urban residents has increased by nearly 60 million a year. By 2050, it is estimated that 70% of the world’s population will be living in cities\(^1\). Urban communication networks and content delivery networks have been introduced to

\(^1\)World population data sheet: http://www.prb.org/
leverage these technologies to better service cities and users alike. Content delivery networks are designed to improve overall network performance by bringing data closer to the geographical locations of users. Urban communication networks have evolved over the years to address urban challenges through the use of information, communication technology and the Internet. Building such a network infrastructure capable of adequately servicing urban locations has become increasingly difficult due to the sheer number of internet devices and users.

Traditionally, content delivery nodes are geographically distributed throughout the world servicing different regions. In large networks such as LinkNYC the same content may be requested by multiple users resulting in the content traversing the entire network multiple times to and from a remote content delivery node hosting the content. This work, however, proposes and analyzes the placement of content delivery cloudlets within LinkNYC’s infrastructure to bring content closer to consumers. This is especially helpful for mobile LinkNYC users naturally experiencing intermittent connectivity as they associate with different cloudlets across a path. Thus, a cloudlet-aware mobile content delivery scheme is proposed to address mobile service continuity issues. Contributions of this work are:

- Performance analysis of the proposed shift of LinkNYC’s infrastructure from a traditional communications network to a content-centric network.

- Establishes that LinkNYC’s communications network vastly improves with not only a content-centric shift but also the placement of multiple content delivery cloudlets geographically distributed throughout LinkNYC.
• Designs and evaluates a cloudlet-aware mobile content delivery scheme for mobile users undergoing frequent handoffs and intermittent connectivity within LinkNYC.

• To our knowledge, this work is the first to analyze and leverage LinkNYC’s urban communications network through content delivery cloudlets improving overall network performance and stability as well as mobile content delivery and service continuity.

The rest of this paper is organized as follows. In Section 4.2 a content-centric LinkNYC infrastructure is introduced and analyzed. A cloudlet-aware mobile content delivery scheme is designed and evaluated in Section 4.3. Finally, the article is concluded in Section 4.4.

4.2 Content-Centric Urban Communication Networks

Coupling urban communication networks with content-centric and delivery principles greatly benefit content producers, consumers and the cities they reside in. Improving their infrastructure using practical approaches to provide more reliable and responsive communications can assist in the technology’s overall success. The underlying concept behind content-centric communication networks is to allow a consumer to focus on the desired named content rather than referencing the physical location or named hosts (IP) where that content is stored. This shift is a product of empirical research resulting in the fact that the vast majority of Internet usage involves data being disseminated from a source to multiple users.
The potential benefits of a content-centric adoption include in-network caching to reduce congestion, improved delivery speeds, simpler network configuration and network security at the data level [6]. This paper combines content-centric and content delivery principles to improve LinkNYC performance and reliability.

4.2.1 Content Delivery Networks

The principle behind content delivery networks (CDN) is to bring data as close to the geographic location of the user as possible to improve overall network performance. This helps eliminate the need to traverse the Internet for content which reduces infrastructure and bandwidth costs while improving network robustness and quality of experience (QoE). For instance, Microsoft Azure’s content delivery network consists of 36 points of presence locations\(^2\) distributed throughout the world as shown in Figure 4.1. However, in addition to being geographically limited, CDN nodes are generally placed at the Internet edges over multiple backbones servicing different regions remotely. Although the concept of bringing content closer to the consumer through caching copies in various geographic locations improves overall network performance; mobile users in large urban communication networks such as LinkNYC naturally endure additional latency due to increased mobility, congestion and hops traversed within the network. This can be improved by placing content even closer to the requesting consumer through content-centric networking and delivery principles and is discussed next.

\(^2\)Figure 4.1 provided by Microsoft Azure: https://azure.microsoft.com/en-us/documentation/articles/cdn-pop-locations
4.2.2 LinkNYC

In November 2014 LinkNYC announced a project plan to provide a first-of-its-kind communications network offering super fast free gigabit Wi-Fi to everyone in New York City through the replacement of thousands of payphones with kiosk-like structures called Links with deployment underway beginning January 2016. Once completed, LinkNYC will be the largest and fastest free public Wi-Fi network in the world. The Links are designed as an update to the standard phone booth and act as Wi-Fi hotspots while also providing basic services such as advertisements, free phone calls, device charging, touchscreen for Internet browsing to access city services, maps and directions. Revenue generated by the Links, through kiosk Ads that are displayed on 55 inch displays, is used to maintain the LinkNYC infrastructure. Each Link is equipped with 802.11ac Wi-Fi technology yielding real world download and upload speeds of 300 and 320 Mbps respectively with an average latency of 5 ms and coverage area of up to 45 meters depending on location. This promising self-contained urban communications network provides
<table>
<thead>
<tr>
<th>Borough</th>
<th># Payphones</th>
<th>Avg. distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhattan</td>
<td>3409</td>
<td>43.2 m</td>
</tr>
<tr>
<td>Queens</td>
<td>1042</td>
<td>136.8 m</td>
</tr>
<tr>
<td>Brooklyn</td>
<td>1004</td>
<td>150.8 m</td>
</tr>
<tr>
<td>Bronx</td>
<td>591</td>
<td>125.5 m</td>
</tr>
<tr>
<td>Staten Island</td>
<td>51</td>
<td>606 m</td>
</tr>
<tr>
<td>Total</td>
<td>6097</td>
<td>212.5 m</td>
</tr>
</tbody>
</table>

Table 4.1: Payphones in the five boroughs of NYC

cities with revenue and analytics while offering consumers free, continuous and reliable connectivity.

In order to leverage content-centric and content delivery networking with LinkNYC's communications infrastructure this work proposes an architectural addition of cloudlets to better service end users. Cloudlets are small-scale cloud datacenters that aim to bring data closer to mobile users and are typically located near the edge of the Internet [7]. Upgrading LinkNYC's Links with cloudlets offers a different dimension to content delivery networking. Naturally, the number of potential cloudlets depends on the number of currently installed payphone locations\(^3\) summarized in Table 4.1. Manhattan is the most dense of the five boroughs with 3,409 payphones and an average distance between them of 43 meters. However, Staten Island is the most sparse with only 51 payphones and an average distance of 606 meters. Unlike traditional content delivery networks, where a limited number of remote servers or nodes are distributed throughout the world, the proposed approach geographically places cloudlets within LinkNYC's large communications

\(^3\)NYC Open Data: https://nycopendata.socrata.com
network. This work proposes the placement of cloudlets in all or a subset of LinkNYC Links. Specific Links are selected as content delivery cloudlets or producers of content and are equipped with an $L2$ storage cache that is much larger than the $L1$ storage cache available on other cloudlets. In order to decide which cloudlets will provide content delivery services a hierarchical clustering technique is used.

![Figure 4.2: Geographic locations of payphones in NYC](image)

4.2.3 Hierarchical Clustering

As consumers become increasingly mobile the placement of content delivery cloudlets within a large network is crucial. Specifically, mobile users that undergo frequent
handoffs as they move across a path results in transport layer issues that reduce service continuity and overall QoE [8]. Placing a single content delivery cloudlet or content producer within the network is insufficient to meet the demand of the mobile consumer. Having content readily available in multiple nearby content delivery cloudlets helps avoid the additional costs of traversing the network for the content. In an effort to analyze LinkNYC’s network and decide for the placement of content delivery cloudlets a hierarchical clustering technique is applied to the borough topologies.

LinkNYC cloudlets are categorized based on their borough as shown in Figure 4.2. Since physical characteristics of New York City’s payphone backhaul connectivity are unknown practical assumptions are made. First, we assume cloudlets are physically connected by fiber optic cables to their nearest neighbor. Given a particular NYC borough, i.e. Brooklyn, a euclidean minimum spanning tree (EMST) is constructed as shown in Figure 4.3 using Prim’s algorithm where edge weights equal the geographic distance between cloudlets. This results in a network topology with \( N - 1 \) edges where \( N \) is the number of cloudlets in a particular borough. Second, content delivery cloudlets are geographically distributed throughout the borough network based on an edge-betweenness hierarchical clustering technique. This technique, known as the GirvanNewman algorithm, progressively removes edges from the original topology that are least central to clusters to form network communities [9]. Edges are ranked based on the number of shortest path combinations that run through them. Higher ranked edges are assumed to be most ”between” communities. The edge with the highest edge-betweenness is re-
moved. Edge-betweenness is then recalculated for the edges that are affected by the removal. This process is repeated until no edges remain resulting in a set of communities. As shown in Figure 4.4, the number of community clusters created for Brooklyn’s topology is 24.

Figure 4.3: Brooklyn’s LinkNYC Network

4.2.4 Content Delivery Cloudlet Placement

Given the resulting cloudlet communities from Section 4.2.3 one node is selected per community as the content delivery or producer cloudlet which stores content in its $L2$ cache. The producer cloudlet is selected based on its distance from the remaining cloudlets within its respective cluster. That is, the node with the minimum sum of shortest paths to the other nodes is selected as the content delivery cloudlet. This ensures content is placed as close to the geographic location of potential consumers within a cluster as possible. Table 4.2 shows the results after
<table>
<thead>
<tr>
<th>Borough</th>
<th>Clusters</th>
<th>CDN %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhattan</td>
<td>64</td>
<td>1.9%</td>
</tr>
<tr>
<td>Queens</td>
<td>37</td>
<td>3.6%</td>
</tr>
<tr>
<td>Brooklyn</td>
<td>24</td>
<td>2.4%</td>
</tr>
<tr>
<td>Bronx</td>
<td>30</td>
<td>5.1%</td>
</tr>
<tr>
<td>Staten Island</td>
<td>9</td>
<td>17.6%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>171</strong></td>
<td><strong>2.8%</strong></td>
</tr>
</tbody>
</table>

Table 4.2: Content delivery cloudlets in each borough

applying this technique to the remaining boroughs. LinkNYC’s overall communications network yields 171 total content delivery cloudlets which makes up 2.8% of the total number of cloudlets within New York City.

4.2.5 Borough Analysis

Different Internet architectures have been introduced for content-centric networking which shifts from the standard IP data packets to Named-Data Networking (NDN). NDN is a future Internet architecture focusing on a content-centric Internet as opposed to today’s host-centric network architecture [6], [10], [11]. These architectures rely on named-content within the Internet to route and direct the flow of data within a network. In these architectures the content is the focus rather than the physical location where the content is stored. ndnSIM is an NDN simulator based on NS-3 and was used to analyze the proposed urban communications network infrastructure.

ndnSIM consists of content consumers and producers. Consumers generate in-
interest requests for specific content chunks whereas producers respond to interest requests with data packets. Every node is equipped with a Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB). When interest packets are received it is first placed into the PIT and the CS is checked for data correlating to this interest. If there is a match the interest request is discarded and the corresponding data packet from the CS is returned. Otherwise the interest packet is forwarded based on information in the FIB. Incoming data packets corresponding to pending interests in the PIT are stored in the node’s CS. Otherwise, the data packet is dropped.

For this analysis a comparison is made between traditional, random and cluster-based content delivery cloudlet placements in Brooklyn which comprises of 1004 nodes. The traditional approach places a single content delivery cloudlet servicing this specific region. In this case, a single cloudlet is randomly chosen to act...
Table 4.3: Brooklyn analysis simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producers</td>
<td>24</td>
</tr>
<tr>
<td>Consumers</td>
<td>60</td>
</tr>
<tr>
<td>L1 Cache</td>
<td>1 GB</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>1 GB</td>
</tr>
<tr>
<td>Requests</td>
<td>100 per second</td>
</tr>
<tr>
<td>Max SeqNo</td>
<td>3000</td>
</tr>
<tr>
<td>MTU</td>
<td>1400 bytes</td>
</tr>
<tr>
<td>Link Rate</td>
<td>10 Mbps</td>
</tr>
<tr>
<td>Latency</td>
<td>5 ms</td>
</tr>
</tbody>
</table>

as a borough’s content delivery cloudlet. This is done with and without intermediate cloudlets caching content to emphasize the improvement in performance when shifting to a content-centric communications network. The random approach randomly selects 24 content delivery cloudlets whereas cluster-based selects one cloudlet per cluster as mentioned in Section 4.2.3. For each simulation 60 consumers are randomly selected and initiate requests for the same content at different time intervals between 0–60 seconds. Results of this analysis are shown in Figure 4.5.

Figure 4.5a shows the network at equilibrium when all consumers have received the final sequence number. This is a promising result as the difference between traditional content delivery placement with and without caching is quite dramatic and emphasizes the improvement of a content-centric shift within LinknNYC’s urban communications network. The average number of hops traveled by the traditional method with caching is slightly over 10 hops compared to over 50
hops for the traditional method without caching. Without caching, content is forced to traverse the entire network to reach the regional content delivery cloudlet resulting in a much higher hop count. In addition, randomly distributing 24 content delivery cloudlets improves the average number of hops to a little under 8. Finally, the cluster-based approach achieves better results yielding on average around 6 hops. Figure 4.5b shows the average hop count midway through the simulation. This is an important distinction to make as it highlights the practical scenarios where, since consumers request content or join/leave the network at different time intervals, higher sequence numbers have yet to be requested and fully disseminate throughout the network. This results in high sequence numbers not being available in cloudlet $L_1$ caches causing intermediate cloudlets to request the content from their neighbors and in turn requiring more hops to traverse.

Figures 4.5d and 4.5c show the total number of cache hits and ratio of hits to misses respectively at the content delivery cloudlets over time. A cache hit occurs when data corresponding to an interest packet is fulfilled from the in-network local cache, $L_1$ or $L_2$, of the cloudlets rather than from the producer application. As more users join the network the cluster-based approach achieves a higher number of cache hits as well as a higher hit to miss ratio for requested content. This is attributed to the cluster-based placement of the content delivery cloudlets as opposed to a random selection. That is, the likelihood that the cluster-based approach would place cloudlets closer to potential consumers is higher than other approaches. Eventually when equilibrium is reached, the cache hit to miss ratios converge for random and cluster-based approaches as higher sequence numbers
Figure 4.5: Content-centric LinkNYC performance results

have fully disseminated the network resulting in higher cache hits. However, traditional methods yield poor results as they experience low cache hit percentages.

This analysis shows that LinkNYC can benefit greatly with not only the adoption of a content-centric infrastructure (traditional with caching) but also with the incorporation of multiple content delivery cloudlets within its urban communications network. Overall a cluster-based placement approach exhibits promising and more stable results yielding lower hop counts and achieving on average a higher rate of cache hits which in turn improves content delivery speeds. In a mobile
environment, where network conditions and service continuity issues are amplified due to mobility, network stability and content delivery speeds are essential to a user’s QoE. Section 4.3 introduces a cloudlet-aware mobile content delivery scheme for LinkNYC’s communications network given a user’s mobility within a cluster serviced by a content delivery cloudlet.

4.3 LinkNYC Mobile Content Delivery

It is clear from Figure 4.2 that New York City is densely equipped with thousands of potential cloudlets within an area populated with multiple mobile users. Within this environment mobile users naturally experience frequent handoffs during a connection lifetime resulting in intermittent connectivity which is detrimental for mobile service continuity and overall QoE [8]. Content interest requests must go through a mobile user’s point of attachment (PoA) or currently connected to cloudlet. This real world scenario risks frequent disconnects and disruptions in a user’s service especially near the edges of a cloudlet’s coverage area. In this case, even though a handoff may be imminent, interest requests must still be requested through the PoA. This causes potential packet losses, increased response times and re-transmissions. Although the placement of content delivery cloudlets improves overall network performance, the physical characteristics and consequences of a mobile user are inevitable. This requires the need to counter these service continuity issues and disruptions by prefetching content on the requesting user’s expected path.
4.3.1 Design

This work uses the architecture from Figure 4.6 to address the mobile issue by prefetching content on candidate cloudlets located on a mobile user’s path. A cloudlet-aware GPS is assumed to provide a mobile user with a path containing multiple cloudlets. In addition, each content delivery cloudlet within a cluster is responsible for maintaining critical cloudlet information such as location, throughput, coverage area and expected traveling speed. This information is shared between the content delivery cloudlets. A cloudlet-aware path $P_N$, where $N$ is the number of cloudlets on a mobile user’s path, entries consist of a 4-tuple, $(C_i, R_i, d_i, s_i)$, representing cloudlet $C_i$ where $R_i$ is the cloudlet throughput, $d_i$ is the cloudlet coverage area in meters and $s_i$ is the expected speed traveled within the cloudlet’s coverage area. In order to prefetch chunks of content Algorithm 6 is applied by a mobile user as it moves within a cluster of cloudlets.
Algorithm 6 Mobile Content Delivery

1: **Input:**
2: $P_N = \{(C_1, R_1, d_1, s_1), ..., (C_N, R_N, d_N, s_N)\}$
3: $M =$ content manifest file
4: $F =$ content size from manifest file
5: **Start:**
6: let $S_{\text{max}} = \left\lceil \frac{F}{MTU} \right\rceil$
7: let $S_{\text{cur}} = 1$
8: let $i =$ index of current cloudlet
9: let $j =$ index of candidate cloudlet
10: let $d_c =$ distance to candidate cloudlet
11: while $j \leq N$ do
12:   if $d_c < \Delta_0 \& F > 0$ then
13:     Get 4-tuple $P_i$ and $P_j$
14:     Calculate content chunk to prefetch
15:     let $E[D_i] = \frac{d_i}{s_i} \times R_i$
16:     let $E[D_p] = \frac{d_j}{s_j} \times R_j$
17:     let $S_{\text{cur}} = \left\lfloor \frac{E[D_i]}{MTU} \right\rfloor + S_{\text{cur}}$
18:     if $j == N$ then
19:       $C_j \rightarrow \text{PreFetch}(M, S_{\text{cur}}, \left\lfloor \frac{F}{MTU} \right\rfloor)$
20:     $E[D_p] = F$
21:   else
22:     $C_j \rightarrow \text{PreFetch}(M ,S_{\text{cur}}, \left\lfloor \frac{E[D_p]}{MTU} \right\rfloor)$
23:   end if
24:   $F = F - E[D_p]$
25:   $i = i + 1$
26:   $j = j + 1$
27: end if
28: end while
First, a mobile user obtains a cloudlet-aware path, $P_N$, through a central server where cloudlet details are maintained while also continuously monitoring its connection rate and speed within the current PoA cloudlet coverage area. Second, a content specific manifest file, which contains content details such as file size, is requested from the cluster content delivery cloudlet. Once received, the mobile user parses the manifest file to acquire the content size and in turn the maximum sequence number, $S_{max}$, based on the maximum transmission unit (MTU) of the network. The mobile user also maintains a current sequence number, $S_{cur}$, which is used to inform candidate cloudlets of the starting sequence number to begin prefetching at as the user moves across a path. Each cloudlet is equipped with a prefetching service which takes as input the content manifest file $M$, $S_{cur}$, and the expected amount to be prefetched, $E[D_p]$. Based on the mobile user’s current speed, $s_i$, distance within the cloudlet’s coverage area, $d_i$, and throughput, $R_i$, the expected amount to be downloaded within the current cloudlet is $E[D_i] = \frac{d_i}{s_i} \times R_i$. Thus, the expected sequence number for the candidate cloudlet to begin prefetching at is $S_{cur} = \left\lfloor \frac{E[D_i]}{MTU} \right\rfloor + S_{cur}$. Information from $P_j$ is then used to calculate the content amount to be prefetched, $E[D_p]$, by the candidate cloudlet, $C_j$. Once a distance threshold, $\Delta_0$, is reached the prefetching service on the candidate cloudlet is initiated. For each iteration, $S_{cur}$ is updated based on the expected amount of content downloaded within the current cloudlet and is the starting sequence number for the candidate cloudlet’s prefetching service. This process is repeated until the entire content has been prefetched or the user has arrived at its destination. Performance results of the proposed mobile content delivery scheme are discussed.
in Section 4.3.2.

4.3.2 Evaluation

To evaluate the performance of the proposed mobile content delivery scheme the overall mobile throughput, packet response times, number of requests re-transmitted and cache hit to cache miss ratios were measured on the simulated cluster topology shown in Figure 4.7. Mobile user $M$ requests 1,800 content chunks per second from the content delivery cloudlet, $C_7$, while moving across a cloudlet-aware path consisting of cloudlets $\{C_1, C_2, C_3, C_4\}$. Table 4.4 details the simulation parameters used.

Figures 4.8a and 4.8b show the throughput and packet response times experienced by the mobile user $M$. The speed of the mobile user varies between 1-2 meters per second in order to simulate a brisk walk in an urban environment. Algorithm 6 is applied as the mobile user moves across its path to proactively request...
Parameter | Value
--- | ---
L1 Cache | 1 GB
L2 Cache | 2 GB
File size | 200 MB
Requests | 1800 per second
MTU | 1449 bytes
$R_i$ | 54 Mbps (802.11a)
$d_i$ | 20 meters
$s_i$ | 1-2 m/s
$\Delta_0$ | 20 meters

Table 4.4: Mobile and cloudlet simulation parameters

(a) Average throughput experienced over time.
(b) Packet response times across path.
(c) Number of re-transmitted requests.
(d) Ratio of cache hits to cache misses.

Figure 4.8: Mobile content delivery performance results.
content in anticipation of service continuity issues and intermittent connectivity. This improves content delivery speeds and minimizes response times as requests are immediately fulfilled by the cloudlet’s $L_1$ caches which contain the prefetched content. Without prefetching sharp drops in throughput are visible when the mobile device associates with $C_2$, $C_3$ and $C_4$. This can cause issues with a user’s QoE especially with time sensitive content such as audio or video. Once the mobile user arrives at its destination the prefetching service is terminated. As shown in Figure 4.8c, prefetching also allows for the mobile device to experience virtually no re-transmissions during its movement across the path, again due to the immediate fulfillment of requests from the $L_1$ caches. This is better illustrated in Figure 4.8d which highlights the ratio of cache hits to cache misses on the cloudlet path. Algorithm 6 ensures content chunks will be available and thus the cache hit ratio will always be 1 within the coverage area of candidate cloudlets. This proactive prefetching technique results in improved delivery speeds, response times and cache hit ratios providing mobile users, with intermittent connectivity, a less strenuous transition which is essential for mobile environments.

4.4 Conclusion

In this paper, a detailed examination of LinkNYC’s urban communications network is performed to investigate the fundamental benefits of shifting from a traditional IP network to a content-centric network where popular content is cached as it disseminates throughout the network. Traditionally, content delivery nodes
are distributed globally to bring content as close to the geographic location of the user. However, promising results show that having multiple content-delivery cloudlets within LinkNYC’s infrastructure dramatically improves overall network performance and stability. Finally, mobile users within LinkNYC are bound to experience intermittent connectivity as multiple cloudlets are encountered. Thus, a cloudlet-aware mobile content delivery scheme is proposed to leverage LinkNYC’s infrastructure and improve a mobile user’s QoE.
Chapter 5: Large Content-Centric Urban Communication Networks
Analysis and Framework Manuscript

Responsive Content-Centric Delivery in Large Urban Communication
Networks: A LinkNYC Use-Case

Hassan Sinky*, Bassem Khalfi*, Bechir Hamdaoui* and Ammar Rayes†

*School of Electrical Engineering and Computer Science, Oregon State University,
Corvallis, Oregon 97331, Email: sinkyh,khalfib,hamdaoub@oregonstate.edu

†Cisco Systems, San Jose, California 95134, rayes@cisco.com

ABSTRACT

Large urban communication networks such as smart cities are an ecosystem of devices and services cooperating to address multiple issues that greatly benefit end users, cities and the environment. LinkNYC is a first-of-its-kind urban communications network aiming to replace all payphones in the five boroughs of New York City (NYC) with kiosk-like structures providing free public Wi-Fi. We consolidate these networks with standalone edge cloud devices known as cloudlets and introduce geographically distributed content delivery cloudlets (CDCs) to store popular
Internet content closer to end users; essential in environments with diverse and dynamic content interests. A content-centric and delivery framework is proposed leveraging NYC’s population densities and CDCs for interest-based in-network caching. Analysis shows that although the adoption of multiple CDCs dramatically improves overall network performance, advanced caching policies are needed when considering increased content heterogeneity. Thus, we propose popularity-driven (pLFU) and cooperation-based (sLFU) caching policies at individual CDCs to account for user and content dynamics over time. The amalgamation of urban population densities, multiple CDC placements and smarter caching techniques helps exploit the ultimate benefits of a content-centric urban communications network and dramatically improves overall network performance and responsiveness. Our proposed solutions are validated using LinkNYC as a use-case.

5.1 Introduction

The proliferation of Internet devices has resulted in efforts to integrate various wireless access technologies for improved performance, increased services and interconnectivity of end users. The recent growth in data demand has prompted researchers to come up with new wireless techniques (e.g., MIMO [1], cooperative communication [2], femtocells [3], etc.) and develop new technologies (e.g., cognitive radio [4], LTE [5], etc.) to be able to meet this high demand. This integration allows for large geographic locations to be serviced providing millions of end users with continuous connectivity and optimal quality of experience (QoE). However,
the world has seen unprecedented urban population growth over the years. In fact, the number of urban residents has increased by nearly 60 million a year. By 2050, it is estimated that 70% of the world’s population will be living in cities\textsuperscript{1}. Urban communication networks and content delivery networks have been introduced to leverage these technologies to better service cities and users alike. Content delivery networks are designed to improve overall network performance by bringing data closer to the geographical locations of users.

Urban communication networks have evolved over the years to address urban challenges through the use of information, communication technology and the Internet. Building such a network infrastructure capable of adequately servicing urban locations has become increasingly difficult due to the sheer number of Internet devices and users (e.g., rapidly increasing numbers of Internet of Things (IoT) devices). Research shows that global mobile data traffic will increase seven-fold reaching 49 exabytes per month by 2021, most of which will be mobile video content, with a percentage projected to reach up to 78% by 2021 \cite{6}. Thus, not only pushing content closer to the end user but also smarter content placement and timely delivery of content are of the utmost importance.

Traditionally, content delivery nodes or datacenters are geographically distributed throughout the world servicing different regions. In large urban networks such as LinkNYC the same content may be requested by multiple users resulting in the content traversing the network multiple times, over large distances, to and from a remote content delivery node hosting the content. Thus, the focus

\textsuperscript{1}World population data sheet: \url{http://www.prb.org/}
of this work is twofold. First, we propose and analyze the placement of multiple content delivery cloudlets (CDC), based on user population densities within a large urban communications network, to bring content closer to consumers. Second, smarter CDC storage techniques are proposed to leverage content-centric in-network caching principles based on content popularity for efficient and timely deliveries. Contributions of this work are as follows:

- Performance analysis of the proposed shift of LinkNYC’s infrastructure from a traditional communications network to a content-centric network.

- Designs and evaluates a CDC placement heuristic which exploits NYC population densities acquired from the 2012 LandScan™ dataset.

- Establishes that large urban communication networks vastly improve with not only a content-centric shift but also the placement of multiple content delivery cloudlets geographically distributed throughout the network.

- Designs and evaluates CDC storage techniques for timely and responsive content delivery within large networks.

- To our knowledge, this work is the first to analyze and leverage LinkNYC’s urban communications network through content delivery cloudlets improving overall network performance and stability as well as mobile content delivery and service continuity.

The rest of this paper is organized as follows. In Section 5.2 content-centric urban communication networks and the LinkNYC infrastructure are introduced,
constructed and analyzed. Next, a CDC placement heuristic is proposed in Section 5.3. Section 5.4 introduces potential content storage solutions for responsive content delivery. Proposed solutions are then analyzed and validated using LinkNYC in Section 5.5. Finally, the article is concluded in Section 5.6.

5.2 Content-Centric Urban Communication Networks

Coupling urban communication networks with content-centric and delivery principles greatly benefit content producers, consumers and the cities they reside in. Improving their infrastructure using practical approaches to provide more reliable and responsive communications can assist in the technology’s overall success. The underlying concept behind content-centric communication networks is to allow a consumer to focus on the desired named content rather than referencing the physical location or named hosts (IP) where that content is stored. This shift is a product of empirical research resulting in the fact that the vast majority of Internet usage involves data being disseminated from a source to multiple users. The potential benefits of a content-centric adoption include in-network caching to reduce congestion, improved delivery speeds, simpler network configuration and network security at the data level [7]. This paper combines content-centric and content delivery principles to improve the performance and reliability of urban networks such as LinkNYC. The principle behind content delivery networks is to bring data as close to the geographic location of the user as possible to improve overall network performance. This helps eliminate the need to traverse the Inter-
net for content which reduces infrastructure and bandwidth costs while improving network robustness and QoE.

5.2.1 Content Delivery Networks

The principle behind content delivery networks (CDN) is to bring data as close to the geographic location of the user as possible to improve overall network performance. This helps eliminate the need to traverse the Internet for content which reduces infrastructure and bandwidth costs while improving network robustness and quality of experience (QoE). For instance, Microsoft Azure’s content delivery network consists of 36 points of presence locations\(^2\) distributed throughout the world as shown in Figure 5.1. However, in addition to being geographically limited, CDN nodes are generally placed at the Internet edges over multiple backbones servicing different regions remotely. Although the concept of bringing content closer to the consumer through caching copies in various geographic locations improves overall network performance; mobile users in large urban networks naturally endure additional latency due to increased mobility, congestion and hops traversed within the network. This can be improved by placing content even closer to the requesting consumer through content-centric networking and delivery principles.

\(^2\)Figure 5.1 provided by Microsoft Azure: https://azure.microsoft.com/en-us/documentation/articles/cdn-pop-locations
5.2.2 Content-Centric Networks

Different Internet architectures have been introduced for content-centric networking which shifts from the standard IP data packets to Named-Data Networking (NDN). NDN is a future Internet architecture focusing on a content-centric Internet as opposed to today’s host-centric network architecture [7], [8], [9]. These architectures rely on named-content within the Internet to route and direct the flow of data within a network. In these architectures the content is the focus rather than the physical location where the content is stored. ndnSIM is an NDN simulator based on NS-3 and was used to analyze the proposed urban communications network infrastructure.

ndnSIM consists of content consumers and producers. Consumers generate interest requests for specific content chunks whereas producers respond to interest requests with data packets. Every node is equipped with a Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB). When in-
interest packets are received it is first placed into the PIT and the CS is checked for data correlating to this interest. If there is a match the interest request is discarded and the corresponding data packet from the CS is returned. Otherwise the interest packet is forwarded based on information in the FIB. Incoming data packets corresponding to pending interests in the PIT are stored in the node’s CS. Otherwise, the data packet is dropped. These concepts are used in designing our responsive content delivery framework for large urban networks. Before presenting our solutions, we first introduce and construct the LinkNYC network which is the basis behind our design choices.

5.2.3 LinkNYC Framework

In November 2014, LinkNYC announced a project plan to provide a first-of-its-kind communications network offering super fast free gigabit Wi-Fi to everyone in New York City (NYC) through the replacement of thousands of payphones with kiosk-like structures called Links. Once completed, LinkNYC will be the largest and fastest free public Wi-Fi network in the world. The Links are designed as an update to the standard phone booth and act as Wi-Fi hotspots while also providing basic services such as advertisements, free phone calls, device charging, touchscreen for Internet browsing to access city services, maps and directions. Revenue generated by the Links, through kiosk Ads that are displayed on 55 inch displays, is used to maintain the LinkNYC infrastructure. Each Link is equipped with 802.11ac Wi-Fi technology yielding real world download and upload speeds of
300 and 320 Mbps respectively with an average latency of 5 ms and coverage area
of up to 45 meters depending on location. This promising self-contained urban
communications network provides cities with revenue and analytics while offering
consumers free, continuous and reliable connectivity.

Figure 5.2: Geographic locations of payphones in NYC

5.2.4 Borough Analysis and Topology Construction

Naturally, the number of potential CDCs depends on the number of currently
installed payphone locations\(^3\) summarized in Table 5.1. Manhattan is the most
dense of the five boroughs with 3,409 payphones and an average distance between
them of 43 meters. On the other hand, Staten Island is the most sparse with only

\(^3\)NYC Open Data: https://nycopendata.socrata.com
LinkNYC cloudlets are categorized based on their borough as shown in Figure 5.2. Since the connectivity of NYC’s payphone backhaul is unknown, we assume that cloudlets are physically connected (e.g., by fiber optic cables) to their nearest neighbors. Given a particular NYC borough (e.g., Brooklyn), we construct an Euclidean minimum spanning tree (EMST) as shown in Figure 5.5b using Prim’s algorithm [10], [11] where edge weights are equal to the geographic distance between cloudlets. EMSTs are useful for telecommunications companies to decide for e.g. where to deploy fiber optic cables considering longer cables are more costly. This results in a network topology with $N - 1$ edges where $N$ is the number of cloudlets in a particular borough. This constructed LinkNYC infrastructure will be used later to validate our proposed solutions.

In order to leverage content-centric and content delivery networking with large urban communications networks such as LinkNYC’s infrastructure, this work proposes an architectural addition of edge cloud devices referred to as content-delivery cloudlets (CDCs) to better service end users. CDCs are small-scale cloud datacenters that aim to bring data closer to mobile users and are typically located near urban areas.
the edge of the Internet [12]. Upgrading, for example, LinkNYC’s Links with these CDCs offers a different dimension to content delivery networking. Unlike traditional content delivery networks, where a limited number of remote servers or nodes are distributed throughout the world, the proposed approach selects and designates a subset of LinkNYC’s links as CDCs (i.e., producers of content). These selected CDCs are each equipped with an \( L2 \) storage cache that is much larger than the \( L1 \) storage cache available on other cloudlets. Content is locally cached as it traverses the network to its destination. For this paper, we assume storage capabilities are limited to CDCs only (i.e. \( L2 \) cache only) as to avoid excessive deployment costs. In order to decide which CDCs will be selected for providing content delivery services a hierarchical clustering technique is applied to the borough topologies based on NYC’s population density distribution detailed in the following section.

5.3 Content Delivery Cloudlet Selection and Placement

As consumers become increasingly mobile with dynamic content interests the placement of CDCs in large cities becomes both crucial and challenging. Specifically, mobile users that encounter multiple cloudlets across a path and undergo frequent handoffs experience service quality continuity issues [13]. With added dynamic content interests, if CDCs’ placement is not carefully designed, these issues may be exacerbated. Relying on a single CDC is insufficient to meet the demand of dynamic consumers, and thus, having content readily available in multiple nearby
CDCs is indispensable to ensure responsive content delivery and maintain good QoE. This avoids the additional costs of traversing the entire network or querying the original publisher for the content. In what follows, we investigate clustering approaches to efficiently select and decide on the placement of multiple CDCs to enable content-centric networking and delivery in smart cities while considering the LinkNYC network as our use-case for applying and evaluating such approaches.

5.3.1 Population Distribution

In order to reflect realistic applications in practice a complete content-centric LinkNYC framework is designed based on NYC’s population density distribution using the United States Department of Energy’s ORNL (Oak Ridge National Laboratory) LandScan™ 2012 dataset. LandScan™ uses spatial data, satellite imagery analysis and a multi-variable dasymetric modeling approach to disaggregate census counts to provide accurate ambient population per square kilometer over a 24 hour period. This results in an average population density during the daytime. Nighttime population densities are acquired using the Socioeconomic Data and Applications Center (SEDAC) Gridded Population of the World (GPW) dataset which uses census and satellite data. Average population densities of LinkNYC are illustrated in Figure 5.4; Note, NYC experiences an influx of 4.5 million people during the day which is mostly concentrated in the Manhattan borough. The datasets were then disaggregated to obtain population density estimates immediately surrounding individual LinkNYC cloudlets. Given the population estimates
a clustering heuristic is applied to each borough for CDC placement.

Figure 5.4: Daytime versus nighttime population densities

5.3.2 CDC Hierarchical Clustering Heuristic

Cluster analysis is an NP-hard problem, thus, efficient heuristic algorithms are commonly employed that converge to a local optimum depending on the application. For this purpose, we propose a population density based clustering heuristic for large urban communication networks and CDC placement.

Initially, cloudlets are part of the same membership and form a single community. In order to introduce content delivery services to the borough network, a cloudlet is chosen as a CDC based on its average hop count to the remaining cloudlets within its respective community. The probability that requests are initiated from each cloudlet, $i$, is assumed to be proportional to its respective surrounding population density, $\gamma_i$, and is defined as $r_i = \frac{\gamma_i}{\sum_{j=1}^{N} \gamma_j}$, where $N$ is the
number of Links in the entire borough network (e.g., Brooklyn). The population densities of all LinkNYC’s Links—γᵢ for Link i—are estimated as described in Section 5.3.1. Let’s now denote by \( S \) the shortest path matrix that contains the length of the shortest path to and from each Link in the borough network. Given the cloudlet request probability vector, \( r = (r_1, r_2, \ldots, r_N_l) \), a weighted average shortest path vector, \( \bar{s} = S \cdot r \), is computed per community and whose entry values represent the weighted average hop count provided by each cloudlet to and from each other. Then, the cloudlet with the minimum sum of weighted average hop counts to remaining cloudlets is selected as the CDC, i.e., \( \text{arg min} \bar{s} \). This ensures content is placed as close to the geographic location of potential consumers within a community as possible. Once selected, the incident edge between the CDC and the cloudlet with the minimum average hop count is removed forming two disjoint communities. Then, for each community, a single cloudlet is selected as a CDC by recomputing vector \( \bar{s} \). This results in a set of \( \bar{s} \)-vectors where the community experiencing a higher average hop count is chosen for additional CDC placement, i.e., \( \text{arg max} \left( \text{min} \bar{s}^{(1)}, \text{min} \bar{s}^{(2)}, \ldots, \text{min} \bar{s}^{(n)} \right) \) where \( n \) represents the current number of communities. Edges are progressively removed from the original topology until no edges remain, resulting thus in a set of communities equal to the number of cloudlets.

The intuition behind this heuristic is to assign more CDCs to highly populated areas. Traditionally, large urban networks rely on a regional CDN (i.e., CDCs = 0) for content caching which provides inadequate responsiveness as content would not only still need to be fetched from the original publisher but also traverse the large
urban network incurring additional latencies. Incorporating in-network caching through the addition of a single CDC immediately improves latency by storing content closer to users. However, we can see from Figure 5.5a that a single CDC still does not facilitate timely content delivery as the average latency to access the CDC is high. To determine an adequate number and initial placement of borough CDCs during the daytime, the elbow in the CDC curve in Figure 5.5a is estimated and used to decide on the number of CDCs to be chosen and are summarized in Table 5.1. Figure 5.5b illustrates the resulting communities and CDCs for Brooklyn’s topology (i.e., CDCs = 25).

<table>
<thead>
<tr>
<th>Borough</th>
<th># Payphones</th>
<th>Avg. distance</th>
<th>CDCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhattan</td>
<td>3409</td>
<td>43.2 m</td>
<td>50</td>
</tr>
<tr>
<td>Queens</td>
<td>1042</td>
<td>136.8 m</td>
<td>25</td>
</tr>
<tr>
<td>Brooklyn</td>
<td>1004</td>
<td>150.8 m</td>
<td>25</td>
</tr>
<tr>
<td>Bronx</td>
<td>591</td>
<td>125.5 m</td>
<td>20</td>
</tr>
<tr>
<td>Staten Island</td>
<td>51</td>
<td>606 m</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>6097</td>
<td>212.5 m</td>
<td>130</td>
</tr>
</tbody>
</table>

Table 5.1: Payphones in the five boroughs of NYC
This analysis shows that large urban networks, such as LinkNYC, can benefit greatly with not only the adoption of a content-centric infrastructure (i.e., single CDC) but also with the incorporation of multiple content delivery cloudlets within its urban communications network. Overall a cluster-based placement approach exhibits promising and more stable results yielding lower hop counts and achieving on average a higher rate of cache hits which in turn improves content de-
livery speeds. Next, we propose content popularity-driven and cooperative caching techniques deployed by CDCs for improved infrastructure performance.

5.4 Popularity-Driven and Cooperative Content-Centric Caching

Internet content of interest to users far surpasses the storage capacity of the CDCs, thereby resulting in frequent content requests from the original content publisher. When content is not stored at intermediate cloudlets, it must be fetched from a remote datacenter incurring additional latency and service continuity issues [14]. Naturally, by increasing cache sizes or the number of CDCs, cloudlets will be able to store and push content closer to end users, but this cannot be done without additional hardware and cache deployment costs. Depending on their needs and resource availability, city officials, policy makers, and network administrators manage to find the balance between acceptable QoE and deployment costs. In this work, we opt for techniques that can improve network performance and QoE while avoiding hardware costs.

5.4.1 Popularity-Driven Content Caching

Traditionally, caching consists of fetching content upon request and storing it locally based on some cache replacement technique, such as First-In-First-Out (FIFO), Random Replacement (RR), Most Recently Used (MRU), Least Recently Used (LRU), Least Frequently Used (LFU), etc [15], in anticipation of future con-
Table 5.2: CDC content store updated per encounter

<table>
<thead>
<tr>
<th>f</th>
<th>cdc</th>
<th>cdc_1</th>
<th>cdc_2</th>
<th>...</th>
<th>cdc_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>f_2</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>f_M</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Window statistics during period $\kappa$

<table>
<thead>
<tr>
<th>f</th>
<th>cdc</th>
<th>cdc_1</th>
<th>cdc_2</th>
<th>...</th>
<th>cdc_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>23</td>
<td>10</td>
<td>...</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>f_2</td>
<td>6</td>
<td>17</td>
<td>...</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>f_M</td>
<td>37</td>
<td>12</td>
<td>...</td>
<td>43</td>
<td></td>
</tr>
</tbody>
</table>

content requests. Current cache information is then used to make replacement decisions (i.e., evicting least used content). LRU is among the most traditional reactive cache replacement policies and is still used in practice today [16]. These traditional solutions, however, are not suitable for Internet content delivery in highly populated cities, such as NYC, mainly due to the diversity, heterogeneity, volume, and dynamic nature of Internet content. In what follows from this section, we propose a content-centric LFU caching method suitable for these highly populated cities that incorporates and relies on the popularity of content encountered by CDCs to decide on which content to be cached.

Our proposed popularity-based content caching method, termed pLFU, works as follows. Each cloudlet computes and maintains an estimate of the average
Figure 5.6: (a) and (b) Content-centric LinkNYC performance results comparing LRU to pLFU considering homogeneous and heterogeneous content request intensity and using different clustering techniques. (c) Average latency as the number of CDCs increases. (d) Average latency as the capacity of CDCs increases.
number, $c_f^{(\kappa)}$, of each content $f$’s requests encountered by the cloudlet at the $\kappa^{th}$ period. We propose to compute $c_f^{(\kappa)}$ as a weighted moving average $\bar{c}_f^{(\kappa)} = \alpha \times c_f^{(\kappa-1)} + (1 - \alpha) \times c_{f,w}^{(\kappa)}$ where $c_{f,w}^{(\kappa)}$ is the number of requests for content $f$ encountered at the $\kappa^{th}$ period during window $w$, and $\alpha$ is a weighting design parameter set between 0 and 1. Examples of a CDCs content store and window statistics are shown in Tables 5.2 and 5.3. The popularity index of content $f$ is then computed as $p_f^{(\kappa)} = \frac{\bar{c}_f^{(\kappa)}}{\sum_{g \in \mathcal{F}} \bar{c}_g^{(\kappa)}}$, where $\mathcal{F}$ is the set of contents the cloudlet has encountered. Subject to cache sizes, CDCs store encountered content with the highest popularity indexes.

Figure 5.6 illustrates the benefit of a content-centric shift within Brooklyn’s LinkNYC network. For this preliminary analysis we leverage Brooklyn’s 25 CDCs, from Table 5.1, where we assume CDCs are capable of storing up to 3% of the network’s total popular content. We also assume the latency required to fetch content from the original content publisher is minimal. Figures 5.6a and 5.6b depict the average latency experienced of LRU and pLFU caching policies while considering content homogeneity and heterogeneity respectively. Although content homogeneity, where all users request the same content through their respective CDCs, is not a realistic case, it does highlight the performance benefit of our population based clustering and CDC placement heuristic from Section 5.3.2, CL, compared to traditional, TR, and k-means, KM, approaches. Since content requests are the same across CDCs, pLFU does not provide any added benefit over LRU. However, when more realistic heterogeneous content requests are introduced we see the performance boost of a pLFU approach. A traditional single regional CDC employing
pLFU (TR\textsubscript{pLFU}) provides a near 25% reduction in average latency per request as opposed to TR\textsubscript{LRU}. Introducing in-network caching to LinkNYC provides great improvements to a traditional infrastructure, however, the average latency remains high (47.5 hops) and is not ideal in dynamically changing environments resulting in sub-par user QoE. Incorporating multiple CDCs within LinkNYC addresses this issue by pushing content closer to the edge. Both KM\textsubscript{LRU} and CL\textsubscript{LRU} provide immediate latency reductions of 58% and 67% compared to TR\textsubscript{LRU}. Although CDCs have limited capacities, content popularity can be learned in anticipation of future requests. Over time, and as content requests intensify, KM\textsubscript{pLFU} and CL\textsubscript{pLFU} exhibit promising results yielding a 65% and 85% latency reduction compared to TR\textsubscript{pLFU}. Using the pLFU policy, Figure 5.6c illustrates average latency as the number of CDCs deployed within Brooklyn’s topology is increased and a content request intensity of $20 \times 10^3$. Naturally, latency improves as more CDCs are deployed, however, the improvement is minimal as the percentage of CDCs reaches around 5%. The effect of increasing CDC cache sizes is shown in Figure 5.6d. Results show that increasing cache sizes alone may not necessarily improve latency since contents still need to traverse to end users. We promote that there exists a balance between the number of CDCs and a CDC’s cache size.

In Sections 5.3 and 5.4.1, a cloudlet-based content placement architecture coupled with a popularity-driven content caching scheme that stores content based on local CDC popularity indexes was proposed. It is shown that integrating content popularity when making content placement and caching decisions reduces both the average downloading time and the network backhaul traffic as opposed to pure
reactive policies. Although the popularity-driven caching scheme, pLFU, yields promising results, it does not guarantee content placement that minimizes the global average latency, as it does not account for content availability and popularity in neighboring CDCs. An amalgam of user mobility, content heterogeneity, and in-network storage introduces several complex challenges. More advanced content-centric solutions must be sought in order to reduce network issues and address underlying limitations such as rising content heterogeneity, user mobility, backhaul load, network failure and deployment costs. Next, we propose a potential solution for overcoming these challenges. Specifically, we propose to exploit and rely on cooperation and information sharing among neighboring CDCs for providing faster content access and lesser backhaul traffic.

5.4.2 Cloudlet Cooperation for Faster Content Access

Content caching and placement decisions should depend not only on local but also neighboring CDC conditions and observations, such as content popularity, storage capacity, content availability in the neighborhood, user population, and link/network condition (congestion, data rates, etc.). Intuitively, when a new content is requested within some local CDC, the decisions on (i) whether to cache the new content or not, (ii) which CDC to cache the content at, and (iii) which existing cache content to evict should all be community-based in that both local and neighboring CDCs should all cooperate and be involved in making such decisions so that globally optimal placements that satisfy users’ QoE while accounting for
resource availability constraints can be made. For example, if the new content is available at a nearby CDC, then there might not be a need for caching it again at the local CDC, thus saving local cache resources yet while still allowing users to receive the content with acceptable latency. Now if the new content is not available locally, nor in neighboring CDCs, then the decision to whether to cache or not should depend on its community popularity, not just its local popularity. If this content is popular enough to cache, then the decision to where it should be cached should weigh in its popularity indexes at the different CDCs within the network. Even if the content is just being requested by a user located within a CDC $i$, it might be more efficient to cache it at a neighboring CDC if future requests are to be generated by users within the neighboring CDC and/or the neighboring CDC has more available cache space.

Designing cooperative content caching and placement approaches that consider the aforementioned performance aspects hasn’t been addressed fully by the research community. And deriving models that capture the various content and network aspects influencing these decisions, such as content popularity, storage availability, content availability, user population, and network condition, is a challenging task that requires careful study. In this work, we propose to introduce a score function $S_{f,i}^{(\kappa)}$ that each CDC $i$ maintains for each of its encountered content $f$, updated every period $\kappa$, and uses to make content placement and caching decisions. Here, $S_{f,i}^{(\kappa)}$ represents the cost associated with caching content $f$ at CDC $i$ at the update period $\kappa$. We propose that this function captures and models the following aspects:

- **Content popularity** ($p_{f,i}^{(\kappa)}$): Popularity index of content $f$ as observed by
CDC $i$ during update window $\kappa$.

- **Content availability** ($a^{(\kappa)}_{f,i}$): An availability binary index of content $f$, where 1 indicates that $f$ is cached in CDC $i$ during update window $\kappa$, and 0 otherwise.

- **Population density** ($r_i$): This reflects the population density of CDC $i$, as described in Section 5.3.1.

- **Inter-CDC delay** ($\ell_{i,j}$): It represents the delay experienced by a user belonging to a CDC $i$ requesting content cached at a neighboring CDC $j$. It essentially captures the number of hops, as well as the link bandwidth capacity of each hop, connecting CDCs $i$ and $j$.

- **Intra-CDC delay** ($\ell_i$): It represents the average delay experienced by a user requesting content from its community CDC $i$.

We propose to model $S^{(\kappa)}_{f,i}$ as a weighted average of a neighborhood score and a local score as such:

$$S^{(\kappa)}_{f,i} = \beta \times S_{\text{neigh}}^{(\kappa)} + (1 - \beta) \times S_{\text{local}}^{(\kappa)}$$

where $S_{\text{neigh}}^{(\kappa)} = \frac{\sum_{j \in \mathcal{N}_i} r_j p^{(\kappa)}_{f,j}}{|\mathcal{N}_i|}$ and $S_{\text{local}}^{(\kappa)} = \frac{r_i p^{(\kappa)}_{f,i}}{\ell_i^{(\kappa)}}$ where $\mathcal{N}_i$ is the set of CDC $i$’s neighboring CDCs and $\beta$ is a design parameter set between 0 and 1. $S^{(\kappa)}_{f,i}$ captures the local and neighborhood benefit, from the perspective of the deciding CDC $i$, of caching content $f$. That is, as content traverses the network each CDC
decides to store the traversing content based on its score value. Considering local and neighborhood characteristics allows for our score function to better service dynamic content environments. Deciding on the number of neighboring CDCs to consider is a design choice discussed in Section 5.5.

Note that additional score functions can be modeled to capture CDC resources (processing, storage, memory, energy, etc.) more accurately and are left for future investigation. The neighborhood score function, $S_{\text{neigh}_{f,i}}$, essentially represents a weighted average latency that user requests generated within CDC $i$ will experience had content $f$ been cached among its neighboring CDCs. $S_{\text{local}_{f,i}}$ represents an average latency if content $f$ is stored locally in CDC $i$. For the extreme case when $\beta = 1$, content resulting in higher potential latencies among our neighbors will be locally stored more often. On the other hand, when $\beta = 0$, our score function stores content with the highest local popularities and low latencies. That is, highly popular content with low latencies to fetch are favored over others.

To assist in the score formulation, we propose the construction of virtual CDC content layers. Assuming boroughs contain a subset of popular Internet content, virtual layers are formed indicating where each content is stored as shown in Figure 5.7. Virtual layers are constructed based on the CSs maintained and periodically shared by CDCs with remaining cloudlets as content traverses the network. Naturally if content is not available within a respective community, it can be requested through virtual CDCs. Otherwise, content is requested from the original publisher. CDCs can use the resulting virtual layers to sift through cloudlets and efficiently compute the neighborhood portion of the score.
5.4.3 Content Popularity Skewness

Naturally, content popularity within communities can vary over time and in order to account for this dynamicity, tuning the design parameter $\beta$, in $S_{f_{i,j}}^{(x)}$, becomes especially useful and can provide a more responsive storage mechanism. We assume content popularity follows the Zipf distribution, 

$$f(\tau, s, M) = \frac{1/\tau^s}{\sum_{m=1}^{M}(1/m^s)}$$

where $\tau$ is a content’s rank in terms of popularity, $M$ is the total number of contents, and $s$ controls the skewness of the distribution. The Zipf distribution is widely used in the literature to describe popularity.

It is clear from Figure 5.8 that as parameter $s$ increases so does the skewness
in content popularity especially when $s \geq 1$. We propose a dynamic parameter $\beta$, in the score function $S_{f,i}^{(\kappa)}$, that is inversely proportional to parameter $s$. That is, for the extreme case when contents have equal popularities (i.e., $s = 0$) simply storing the most popular content locally, as done with pLFU, will not suffice. However, querying content popularities and availabilities among neighboring CDCs for storage decisions intuitively improves responsiveness by effectively consolidating storage capabilities. Thus, a higher $\beta$ value is preferred to favor our neighborhood in our cache decisions. Conversely, when content skewness is high (i.e., $s = 2$), it is more efficient to locally store the most popular content within a community. In other words, as content popularities are more skewed our score function favors storing the most popular content locally. For this reason a dynamic $\beta$ parameter is necessary to balance between local and neighborhood based storage decisions depending on the $s$ parameter in the Zipf distribution. This allows $S_{f,i}^{(\kappa)}$ to be adjusted accordingly and adapt to network changes over time.

![Figure 5.8: Popularity indexes as a function of $s$](image)
5.4.4 Cooperative Content Placement

The parameter $\beta$ allows us to balance between the need for having responsive content delivery locally versus among our CDC neighbors. This is especially useful for dynamic content popularity within large urban communication networks. As these above network and content conditions change over time, each CDC must periodically maintain and compute score function values for encountered contents. This could be done by having CDCs query neighboring CDCs for content popularity indexes, population densities, CDC latency and resource availability, and use this information for updating these values, which are then used as follows for content placement decisions:

- Content popularity indexes, $p_{f,i}^{(\kappa)}$, are maintained for each encountered content $f$ during period $\kappa$ as described in Section 5.4.1
- For each content $f$ encountered at CDC $i$, the $S_{neigh, f,i}^{(\kappa)}$ and $S_{local, f,i}^{(\kappa)}$ portions of the content score value, $S_{f,i}^{(\kappa)}$, are computed and maintained during period $\kappa$
- For each CDC $j$ in our set of neighbors $\in \mathcal{N}$, inter-CDC latency, $\ell_{i,j}^{(\kappa)}$, is computed based on content $f$’s availability among our neighbors. Note, the additional latency required to fetch content from the original publisher is also captured:
\[
\ell_{i,j}^{(\kappa)} = \begin{cases} 
\ell_{i,j} + \ell_{j,\text{origin}}^{(\kappa)} & a_{f,i}^{(\kappa)} = 0 \\
\ell_{i,j} & a_{f,i}^{(\kappa)} = 1 
\end{cases}
\]

- Intra-CDC latency, \( \ell_i^{(\kappa)} \), is computed during period \( \kappa \) as the average latency (i.e., hops) from each cloudlet to its community CDC \( i \). Similarly, in the event that \( a_{f,i}^{(\kappa)} = 0 \), \( \ell_{j,\text{origin}}^{(\kappa)} \) is accounted for to capture the additional latency required to fetch content from the original publisher.

- Design parameter \( \beta \) is set based on an estimation of the \( s \) during period \( \kappa \) and maintained per CDC.

- For each content \( f \) encountered at CDC \( i \), it is stored locally if \( f \in S_{\text{max}} \) where \( S_{\text{max}} \) is the set of contents with the highest local content scores, subject to CDC storage capacity. Otherwise no caching takes place. If CDC \( i \)'s storage is full and \( f \in S_{\text{max}} \) then the content with the minimum score value is evicted from the local cache.

Considering both local and neighborhood conditions and content availability in modeling \( S_{f,i}^{(\kappa)} \) allows us to account for changing content popularities within different communities over time. Our intuition indicates that storage decisions must rely heavily on a CDC’s content popularity distribution. That is, as content popularity becomes more uniform, considering content availability among our neighbors improves performance. However, as content popularity is more skewed, storage decisions should be based on local CDC content popularities. We discuss
our design choices and analyze our results next.

5.5 Framework Analysis: A LinkNYC Use-Case

A detailed examination using LinkNYC’s urban communications network is performed to investigate the fundamental benefits of shifting from a traditional IP network to a content-centric network where popular content is cached as it disseminates throughout the network. Traditionally, content delivery nodes are distributed globally to bring content as close to the geographic location of the user. However, promising results show that having multiple content-delivery cloudlets within LinkNYC’s infrastructure coupled with smarter caching techniques dramatically improves overall network performance and stability. In an increasingly mobile and dynamic environment, where network conditions and service continuity issues are amplified due to user mobility and varying content interests; network stability, efficiency and content delivery responsiveness are essential to a user’s QoE.

5.5.1 Setup

In this analysis we focus on NYC’s Brooklyn borough. Among the topology’s 1004 cloudlets, 25 CDCs are geographically selected based on population densities using our technique from Section 5.4. Content requests are then generated following a Zipf distribution with varying $s$ parameters per community to realistically sim-
ulate community-specific interests and changing content popularity. We assume LinkNYC’s popular content library, $M$, contains 600 contents categorized by type (i.e., Sports, Entertainment, Politics and Education) where each CDC’s storage capacity, $C_i$, is set to 20 contents. Requesting content from its original publisher also yields varying latencies per community (i.e., between 250 and 500 hops). In addition, popularity skewness and interests are periodically shuffled among communities to capture time variation within the LinkNYC infrastructure. Finally, design parameters $\alpha$ and $\beta$ are chosen to control (i) the framework’s sensitivity to popularity changes and (ii) the degree to which we balance between neighborhood and local content storage decisions respectively. Our simulation parameters are summarized in Table 5.4.

5.5.2 CDC Neighborhood Size

 Neighborhood size can play an important role in overall network performance. Increasing the size of CDC $i$’s neighborhood, $N_i$, improves performance by effectively consolidating CDC capabilities. Figure 5.9a emphasizes this performance increase with respect to latency as the neighborhood size is grown to 24. This allows for efficient cooperation among CDCs when making content storage decisions. Since the number of CDCs are relatively low, the negative impact of increasing neighborhood size is often negligible. However, in networks comprising of hundreds of thousands or even millions of devices, such as Massive Internet of Things (MIoT) networks, increasing the neighborhood size can be disadvantageous in terms of
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borough</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Cloudlets</td>
<td>1004</td>
</tr>
<tr>
<td>CDCs</td>
<td>25</td>
</tr>
<tr>
<td>Content Categories</td>
<td>Sports, Education, Politics, Movies</td>
</tr>
<tr>
<td>$</td>
<td>M</td>
</tr>
<tr>
<td>$C_i$</td>
<td>20</td>
</tr>
<tr>
<td>Number of requests</td>
<td>$10^6$</td>
</tr>
<tr>
<td>$\ell_{\text{origin}}$</td>
<td>[250, 500] hops</td>
</tr>
<tr>
<td>$</td>
<td>N_i</td>
</tr>
<tr>
<td>$s$ range</td>
<td>[0, 2]</td>
</tr>
<tr>
<td>$\hat{s}$ observations</td>
<td>$10^3$</td>
</tr>
<tr>
<td>$w$</td>
<td>$10^2$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>.2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$1 - \frac{1}{1+e^{-20(s-0.5)}}$</td>
</tr>
</tbody>
</table>

Table 5.4: LinkNYC Simulation parameters
complexity due to the added overhead injected into the infrastructure through the exchange of content stores and popularity indexes [17]. LinkNYC network administrators should decide on appropriate neighborhood sizes depending on the application. For instance, during high profile sporting events or tense political climates, many contents may be popular throughout the entire NYC borough and thus using a neighborhood size equal to the total number of CDCs may be preferred. Conversely, geographically restricted interests such as social events may cause only one or two communities to share common interests and thus smaller or even dynamic neighborhood sizes may be beneficial to the overall performance of the network.
Figure 5.9: (a) sLFU performance as CDC neighborhood size is increased. (b) Comparing traditional cache replacement policies with pLFU and sLFU considering static content popularity skewness.

5.5.3 Impact of Content Popularity

Figure 5.9b illustrates the effect content popularity has within LinkNYC. We compare our proposed techniques, pLFU and sLFU, with traditional LRU and LFU cache replacement policies. Here we assign the same $s$ value to all Brooklyn communities to emphasize the importance of tuning the design parameter $\beta$ in $S_{f,i}^{(\kappa)}$. 
based on the content distribution. With $s$ values close to zero, $\beta$ is adjusted accordingly allowing for sLFU to cooperate more with neighboring CDCs to make storage decisions, drastically reducing latency by up to 33% compared to LFU and pLFU, and up to 43% compared to LRU. However, as we increase $s$, and in turn content popularity skewness, sLFU appropriately relies more on local CDC popularity indexes for storage decisions. That is, when content popularity is very skewed (i.e. $s \geq 2$), it is sufficient to only store the most popular content locally; hence, traditional methods also perform well.

5.5.4 Content Dynamicity and Estimation

Figure 5.10 introduces content dynamicity within our content-centric LinkNYC network. Content popularity skewness and interests are periodically shuffled among communities to capture time and content variation. This depicts shifting user interests per community over time (i.e. different sporting events, changing political climates, etc.). During period $\kappa$, where $w$ contents have been encountered, popularity indexes can be leveraged to estimate the $s$ parameter of the content distributions. Each CDC $i$ can then incorporate its estimated $s$ value for tuning design parameter $\beta$ of the score function $S_{f,i}^{(\kappa)}$. We use maximum likelihood estimation (MLE) to find $s$ based on our current content observations (i.e. $10^3$ observations) allowing $\widehat{s}$LFU to adapt to changing popularities over time.

Figures 5.10a and 5.10b depict LinkNYC performance as content requests intensify and while popularity skewness is relatively low. The LRU cache replacement
Figure 5.10: (a), (b), and (c) Illustrate content-centric LinkNYC performance results comparing LRU, LFU, pLFU and sLFU considering a dynamic network with periodically fluctuating content interests and popularities where $s$ is changed per community with a range between 0 and 2. (d) The ratio of cache hits to cache misses with respect to $s$. 
policy is the worst performer and results in high average latency, over 200 hops, that is not suitable for a dynamic and mobile environment. LFU and pLFU perform similarly, however, pLFU is able to converge quicker to an average latency of nearly 180 hops. sLFU’s and ñLFU’s cooperation with neighboring CDCs allows them to perform noticeably better achieving over a 30% improvement compared to traditional methods even in less than ideal conditions where content popularity is relatively similar across communities. When maximum content popularity skewness is increased (i.e. $s = 2$), as shown in Figure 5.10c, traditional methods perform better but are still no match for sLFU and ñLFU which still provide a 30% reduction in average latency. Figure 5.10d illustrates the cache hit percentages when different $s$ parameters are used. sLFU consistently achieves a high cache hit percentage than traditional methods constantly achieving a near 80% hit rate even with low $s$ values. sLFU’s cooperation-based technique is a novel approach for content placement in large urban communication networks. Implementing sLFU at CDCs provides an efficient and cooperative LinkNYC infrastructure for responsive content delivery in dynamically changing environments; essential for mobile users where timely delivery is of the essence.

5.5.5 User Mobility

Mobile users within LinkNYC are bound to experience intermittent connectivity as multiple cloudlets are encountered risking frequent disconnects and disruptions in a user’s service especially near the edges of a cloudlet’s coverage area [13]. This
causes potential packet losses, increased response times and re-transmissions. User mobility and dynamic content interests introduce added complexity to an already challenging issue. In order to accommodate mobile users as well as avoid interfering with popular community content we propose the consolidation of CDC storage with a smaller, temporary $L_1$ storage strictly used to store prefetched mobile user content on candidate CDCs located on a mobile user’s path. Once a mobile user consumes the prefetched content the $L_1$ cache is immediately vacated in preparation for other mobile users. In our previous work, we designed a cloudlet-aware mobility based solution to address service continuity issues as multiple cloudlets are encountered across a path. In [14] a proactive algorithm is designed and applied by a mobile user as it moves within a community of cloudlets to prefetch content chunks. Prefetching content on candidate CDCs or cloudlets across a path has been shown to address mobility induced service continuity issues.

5.6 Conclusion and Future Directions

In this article, we investigated the fundamental benefits of shifting a large urban communications network to a content-centric network where popular Internet content is cached as it disseminates throughout the network. Different caching techniques are designed and evaluated. Initially, a technique to learn and store the most popular content locally at individual CDCs is proposed. However, results show that performance can be improved through cooperative interaction among neighboring CDCs in order to consolidate storage capabilities. Promising results
show that an amalgamation of in-network caching, the placement of multiple CDCs and advanced storage techniques employed within an urban network’s infrastructure dramatically improves overall network performance and responsiveness. This type of network unlocks a variety of applications and open challenges. An economical aspect can be introduced where CDC storage space is rented to various businesses to serve popular Internet content or customer services while providing an additional source of revenue for the network. Moreover, LinkNYC can be exploited through mobile cloud computing by augmenting CDC computing capabilities. The focus is still latency minimization while allowing users to offload heavily computational tasks. Our framework lays the foundation for a first-of-its-kind content-centric urban communication network with the focus of minimizing latency and leveraging interest-based content caching through popularity learning and cooperative content placement.
Chapter 6: Conclusion and Future Work

The proliferation of mobile users, portable handheld devices and vast Internet content have become an enabling factor for wireless communication systems and responsive content delivery solutions. QoEThe contributions of this dissertation are threefold. First, we analyzed TCP’s drawbacks as a result of handovers between networks with varying data rates. A list of service continuity issues are categorized and presented. Modifications to TCP were proposed based on the bandwidth delay product and cross-layer assistance. Results show that packet jitter, round-trip times, queue sizes and network stability are reduced significantly providing improved performance in handoff scenarios drastically reducing user perceived disruptions and glitches.

Second, a promising new transport layer protocol, MPTCP, is analyzed and amended. MPTCP uses multiple TCP subflows to maintain a network connection even when connection endpoints change. Modifications to MPTCP are needed to address service continuity issues resulting from increased user mobility. Our system model addresses these issues by proactively adjusting MPTCP subflows in anticipation of potential handoffs along a mobile user’s path. Results show overall QoE improvements in round-trip times, jitter and throughput. Our low-level transport and physical layer solutions help lessen mobility induced service continuity issues. However, empirical research shows that relying on transport
layer protocols alone does not suffice for an improved user experience. Fundamental changes to network infrastructure are needed for optimal QoE.

Finally, we investigate the fundamental benefits of shifting a large urban communications network to a content-centric network where popular Internet content is cached as it disseminates throughout the network. Different caching techniques are designed and evaluated. Initially, a technique to learn and store the most popular content locally at individual CDCs is proposed. However, results show that performance can be improved through cooperative interaction among neighboring CDCs in order to consolidate storage capabilities. Promising results show that an amalgamation of in-network caching, the placement of multiple CDCs and advanced storage techniques employed within an urban network’s infrastructure dramatically improves overall network performance and responsiveness.

This type of network unlocks a variety of applications and open challenges. An economical aspect can be introduced where CDC storage space is rented to various businesses to serve popular Internet content or customer services while providing an additional source of revenue for the network. Moreover, LinkNYC can be exploited through mobile cloud computing by augmenting CDC computing capabilities. The focus is still latency minimization while allowing users to offload heavily computational tasks. Our proposed framework lays the foundation for a first-of-its-kind content-centric urban communication network with the focus of minimizing latency and leveraging interest-based content caching through popularity learning and cooperative content placement.
Chapter 1: Introduction


Chapter 2: TCP-CLAH Manuscript


Chapter 3: CLA-MPTCP Analysis and Framework Manuscript


Chapter 4: Large Content-Centric Urban Communication Networks

Proof of Concept Manuscript


Chapter 5: Large Content-Centric Urban Communication Networks
Analysis and Framework Manuscript

support in multi-hop wireless networks with MIMO links”, Selected areas in

in wireless networks: Performance analysis and optimum power allocation”,

analysis of two-tier femtocell networks”, Vehicular technology, ieee transac-

sharing through reinforcement learning”, Vehicular technology, ieee transac-

lte uplink with synchronous harq constraints”, Ieee transactions on mobile
computing, no. 99, pp. 1–1, 2014.

Cisco white paper, 2017.

Papadopoulos, L. Wang, and B. Zhang, “Named data networking”, Sigcomm


