Content Replication and Placement Schemes for Wireless Mesh Networks

by

Zakwan Al-Arnaout

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2014

Abstract

Recently, Wireless Mesh Networks (WMNs) have attracted much of interest from both academia and industry, due to their potential to provide an alternative broadband wireless Internet connectivity. However, due to different reasons such as multi-hop forwarding and the dynamic wireless link characteristics, the performance of current WMNs is rather low when clients are soliciting Web contents. Due to the evolution of advanced mobile computing devices; it is anticipated that the demand for bandwidth-onerous popular content (especially multimedia content) in WMNs will dramatically increase in the coming future.

Content replication is a popular approach for outsourcing content on behalf of the origin content provider. This area has been well explored in the context of the wired Internet, but has received comparatively less attention from the research community when it comes to WMNs. There are a number of replica placement algorithms that are specifically designed for the Internet. But they do not consider the special features of wireless networks such as insufficient bandwidth, low server capacity, contention to access the wireless medium, etc.

This thesis studies the technical challenges encountered when transforming the traditional model of multi-hop WMNs from an access network into a content network. We advance the thesis that support from packet relaying mesh routers to act as replica servers for popular content such as media streaming, results in significant performance improvement. Such support from infrastructure mesh routers benefits from knowledge of the underlying network topology (i.e., information about the physical connections between network nodes is available

at mesh routers).

The utilization of cross-layer information from lower layers opens the door to developing efficient replication schemes that account for the specific features of WMNs (e.g., contention between the nodes to access the wireless medium and traffic interference). Moreover, this can benefit from the underutilized resources (e.g., storage and bandwidth) at mesh routers. This utilization enables those infrastructure nodes to participate in content distribution and play the role of replica servers.

In this thesis, our main contribution is the design of two lightweight, distributed, and scalable object replication schemes for WMNs. The first scheme follows a hierarchical approach, while the second scheme follows a flat one. The challenge is to replicate content as close as possible to the requesting clients and thus, reduce the access latency per object, while minimizing the number of replicas. The two schemes aim to address the questions of where and how many replicas should be placed in the WMN. In our schemes, we consider the underlying topology joint with link-quality metrics to improve the quality of experience. We show using simulation tests that the schemes significantly enhance the performance of a WMN in terms of reducing the access cost, bandwidth consumption and computation/communication cost.

To my dearest wife Ajnur,
who made all of this possible with her
endless support, patience and encouragement.

Papers Published/Accepted

Journal Publications

- Zakwan Al-Arnaout, Qiang Fu, and Marcus Frean. "A Divide-and-conquer Approach for Content Replication in WMNs," in *The International Journal of Computer and Telecommunication Networks* (COMNET), Elsevier, vol. 57, no. 18, pp. 39143928, 2013

Conference Publications

- Zakwan Al-Arnaout, Qiang Fu, and Marcus Frean. "On the Placement of Web Content Replicas in WMNs," accepted for publication in *IEEE International Conference on Communications*, ICC'14, 10-14 Jun. 2014, Sydney, Australia.

- Zakwan Al-Arnaout, Qiang Fu, and Marcus Frean. "Exploiting Graph Partitioning for Hierarchical Replica Placement in WMNs," in *Proceedings of the 16th ACM/IEEE International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM'13, 3-8 Nov. 2013, Barcelona, Spain.

- Zakwan Al-Arnaout, Jonathan Hart, Qiang Fu, Marcus Frean. "Link-Quality Aware Object Replication and Placement for Multi-hop Wireless

Mesh Networks," in *Proceedings of the 21st IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS'13, 14-16 Aug. 2013, San Francisco, USA.

- Zakwan Al-Arnaout, Jonathan Hart, Qiang Fu, and Marcus Frean. "MP-DNA: A Novel Distributed Replica Placement Heuristic for WMNs," in *Proceedings of the 37th Annual IEEE Conference on Local Computer Networks*, LCN'12, 22-25 Oct. 2012, Clearwater Beach, FL, USA.

- Zakwan Al-Arnaout, Qiang Fu, and Marcus Frean. "A Content Replication Scheme for Wireless Mesh Networks," in *Proceedings of the 22nd ACM international workshop on Network and Operating System Support for Digital Audio and Video*, NOSSDAV'12, 7-8 Jun. 2012, Toronto, ON, Canada.

- Zakwan Al-Arnaout, Qiang Fu, and Marcus Frean. "A Novel Distributed Content Replication and Placement Scheme for Wireless Mesh Networks," in *Australasian Telecommunication Networks and Applications Conference*, ATNAC'11, 9-11 Nov. 2011, Melbourne, Australia.

Acknowledgments

The three years and a half spent working on my Ph.D. thesis at the School of Engineering and Computer Science is an experience that I will never forget. This thesis would have never been completed without the help and support of many persons who contributed to it directly or indirectly.

First, I would like to thank Dr. Qiang Fu for giving me the opportunity to pursue my Ph.D. studies under his supervision, and for providing me with excellent support and dedication during the degree period. I am privileged for having him as my supervisor for his kindness in helping me to find my research direction and giving the hints on how to do research. Furthermore, I would like to thank my secondary supervisor, Dr. Marcus Frean, for his support during my study at Victoria University of Wellington.

Special thanks to Victoria University of Wellington for granting me a scholarship, which made it possible for me to pursue my degree. Next, I want to thank the members of my thesis examination committee: Dr. Bjorn Landfeldt, Dr. Marius Portmann, and Andy Linton for the considerable amount of work, which requires reading the full document and looking for problems and potential enhancements with a critical mind. I would like to thank them for their precious time and constructive comments on this thesis.

I am very grateful for having the opportunity to work with so many wonderful people at the Networking Lab, these people deserve credit for providing such

a nice and friendly work environment. My colleagues, Dong, Jonathan, Brett, and Ihab. It has been a privilege to work with all of you.

Last but not the least, this thesis is dedicated to my wife, Ajnur Miftari, my kids Najmeddin, Luljeta & Hajar, and my parents Shuaib & Ilham for supporting and encouraging me to pursue this degree.

Contents

List of Figures

List of Tables

List of Acronyms

CM Content Manager

CDN Content Delivery Network

D-EWMA Double Exponentially Weighted Moving Average

DBT Delegates Binary Tree

DHT Distributed Hash Table

DN Delegate Node

DNS Domain Name System

ETT Expected Transmission Time

ETX Expected Transmission Count

FTP File Transfer Protocol

IGW Internet Gateway

IP Internet Protocol

LP Linear Programming

LRU Least Recently Used

MAC Media Access Control

MANET Mobile Ad Hoc Network

MC Mesh Client

ML Minimum Loss

MP-DNA Multiple Partition per Delegate Node Assignment

MR Mesh Router

OLSR Optimized Link State Routing Protocol

P2P Peer-to-Peer

RL Replica List

SP-DNA Single Partition per Delegate Node Assignment

TCP Transmission Control Protocol

WMN Wireless Mesh Network

Chapter 1

Introduction

The technological revolutions in the last few years have emerged new forms of collaboration and interaction between community members. This type of networking is known as community networks as they share the same interests and collaborate to satisfy common objectives. At the application layer [1], such trends have been served by technologies like Web 2.0, social networks, mobile computing, etc. This revolution has evolved to new forms of resource sharing and building new distributed networks infrastructures known as community networks that are available to serve the community members.

Recently, many initiatives were exploiting the potentials of Wi-Fi technology to offer ubiquitous Internet access, in either free or commercial wireless hotspots. However, the cost of the wired infrastructure combined with Wi-Fi's small transmission range; make it difficult to cover wide areas by means of only hotspots. Wireless Mesh Networks (WMNs) provide smart means to reduce such costs, since only some of the mesh routers that form the network should have a direct connection to the Internet. MIT Roofnet [2], Rice University's TAP [3], Athens Wireless [4], and Berlin Freifunk [5] are representative examples for such initiatives. Such wireless infrastructures can provide connectivity with notably

less cost than the wired solutions in many cases.

Apart from providing a wireless one-hop link towards the Internet, users can form WMNs with their own wireless access points utilizing the large amount of underutilized connectivity that can facilitate access to the Internet. However, this is not the only potential a WMN can provide us. Other potential services that might increase the network capacity can be achieved by enabling resource sharing such as content sharing, caching, services for mobile users, P2P applications, VoIP, online games, IPTV, live streaming, and FTP and Web access.

The incremental deployment of community WMNs demands for improving WMN performance, since it will have an influence on the emerging number of users. A significant problem that arises as the Internet traffic flows through the limited number of gateways (one or more but limited) is the heavy congestion around the gateway. Previous research [6, 7] has shown that significant workload locality exists in Internet content retrieval from a given population of clients. Locality in a workload means that multiple users request the same content over time or that multiple users of the network request the same content at the same time. Web caching was introduced and broadly studied to exploit locality of workload aiming to reduce the Internet-access traffic and the user's perceived access latency. As many WMNs are used to provide Internet access, the workload locality also transfers into WMNs [8], which provides an evidence of workload locality in Internet access for client population normally found in a WMN.

In the following sections, we will give the reader a brief background about the different approaches for content delivery, a classification for content outsourcing policies, a comparison between replication & caching systems, and the architecture of WMNs. Then, we describe our research motivation, research goals, research contributions, and the thesis organization.

1.1 Approaches Used for Content Delivery

Generally, there are five approaches for content delivery that we briefly describe as follows:

1. **Traditional single server**: This approach is the easiest to implement, where a single origin server handles content delivery. This is the traditional way used in the 1990's decade by Web and FTP servers. However, this approach is not scalable and failed in the face of enormous flash crowd events such as the 9/11 events in USA.

2. **IP multicasting**: This refers to the delivery of content to a group of clients simultaneously in a single transmission from the source and while en route; packets are duplicated by network elements (i.e., routers) for the purpose of relaying the duplicate packets to one of the member clients served by the network element. IP multicasting reduces both server and network load. However, it requires synchronous receivers (i.e., receivers request the same content at the same time). Furthermore, it is not widely deployed and does not address the problem of high access latency.

3. **Traditional Web caching and prefetching**: Web caching [9, 10] offers advantages such as reduced network traffic and short access latency. However, it has drawbacks such as small hit rates –due to the dynamic and rapid replacement– and low latency reduction of 26%, to overcome such problems, traditional caching is coupled with prefetching to predict future requests for Web objects and fetching them into the cache before being requested. Coupled caching/prefetching provides at best 60% latency reduction. However, it does not improve availability during flash crowds and is still limited in its ability to reduce latency.

4. **Content Delivery Network**: As a reaction to the flash crowds events, the research community has put more research efforts on the area of Content Delivery Networks (CDNs), where a CDN such as Akamai [11] and Limelight Networks [12], act on behalf of the content provider by providing a platform for accelerating content delivery. A general architecture (as depicted in Fig. 1.1) of a CDN system involves four main components [13]:

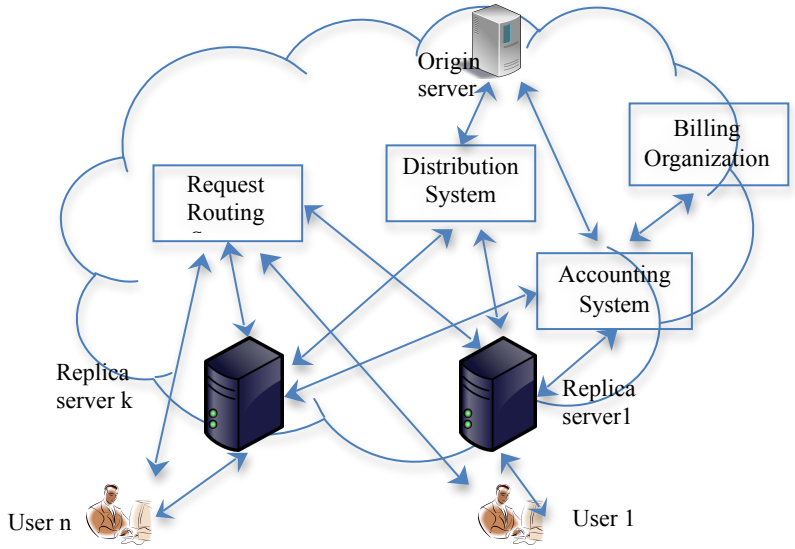


Figure 1.1: A typical CDN architecture.

(a) **The content delivery component**: This includes the origin server and a number of replica servers (or surrogates) that deliver copies of content to the end users; these replica servers are placed in locations close to the clients.

(b) **The request routing component**: This component redirects clients' requests to the appropriate replica servers; it also maintains an up-to-date view of the contents stored in the replica servers by communication with the distribution component. Typically, CDNs use DNS redirection [14], where the CDN exploits DNS servers and the clients

are redirected to the content server based on their DNS queries.

(c) **The distribution component**: This component moves content from the origin server to the replica servers to maintain consistency of content.

(d) **The accounting component**: This component maintains logs of client accesses and records the usage of the CDN servers.

5. **Content delivery in P2P networks**: This approach is especially used for delivering large files, where the file to be delivered is available from a server known as seed. Furthermore, there is a tracker server that keeps track of all the clients in the network. To download a file, a client contacts the tracker inquiring about the needed file, and then the tracker sends a list of peers who are currently downloading that file or possess all of it. Then the client selects some peers from the set and starts downloading chunks from them. The client tries to find the best peers to download the chunks by trying different peers from the peers set. A variety of approaches [15] exist to map content availability in P2P networks such as Centralized directory (e.g., Napster [16]), Distributed Hash Tables (DHT) (e.g., Chord [17]) and Flooded request (e.g., Gnutella [18]).

1.2 Content Outsourcing Policies

Given a set of replica servers in a CDN infrastructure and content to be delivered, choosing an efficient content outsourcing policy is important. Content outsourcing policies are classified [19] into four distinct categories:

1. **Uncooperative pull-based**: Clients' requests are directed to their closest surrogate server (e.g., geographic proximity). The shortcomings observed

are: (*i*) Server selection is not always optimal from which to serve the content; and (*ii*) Incur excessive replication redundancy.

2. **Cooperative pull-based**: Clients' requests are directed (DNS redirection) to their closest surrogate server. The main benefit is that the surrogate servers cooperate with each other in case of cache misses (using a distributed index), they find nearby copies of the requested objects, and store them in their caches. This scheme is reactive; hence, incurs a large communication overhead when the number of clients is large. Moreover, it does not offer high reliability when the content changes rapidly or when the coherency requirements are strict.

3. **Uncooperative push-based**: Content is pushed (proactively) from the origin server to the surrogate servers. The requests are satisfied either at a local surrogate server or at the origin server. As a result, this scheme does not have much flexibility in adjusting replication and management cost.

4. **Cooperative push-based**: Content is pushed (proactively) from the origin server to the surrogate servers. The request is served locally if the surrogate server has the replica; otherwise, it forwards the request to the closest server that has the replica. However, if the requested object has not been replicated/outsourced by some surrogate server, then the origin server serves the request. This replication mechanism is known as long-term prefetching and works by identifying collections of valuable objects to replicate. Although it incurs communication and management cost, but it benefits from the efficient sharing of bandwidth among the surrogate servers and also reduces the replication redundancy, which reduces the cache consistency maintenance costs.

1.3 Replication versus Caching

Caching and replication are used in a mixed environment and the discussion of one is not complete without discussing the other. In content networking, we have two different approaches to create replicas of objects in different network nodes. Although the two approaches do the same task and both terms are used interchangeably in the literature, but there are a number of differences between them that we list below:

1. Replication is performed proactively (push-based) by distributing object replicas to the replica servers, whereas caching is performed reactively (pull-based) as a result of query execution sent by a client node that caches the query result in its cache space if the cache can accommodate it.

2. Replication arises at servers even if the content was not solicited from these servers. On the contrary, the cache will remain empty if clients have issued no queries. As a result, caching decision is made by the query process, while replication decision is made by a separate component at every server and is independent of the query process.

3. In replication, object replicas remain stored in the servers until they are explicitly evicted, whereas object copies are stored in cache servers until they are replaced by other object copies using a replacement policy such as –but not limited to– the Least Recently Used (LRU) policy, Least Frequently Used (LFU) policy or until they are evicted from the cache when it becomes invalid (stale).

4. Replication schemes provide a strong consistency model for object replicas and accessible at servers at all times, while caching maintains the consistency by using mechanisms based on invalidation and removing

stale copies. Table 1.1 summarizes the differences between caching and replication.

Table 1.1: A comparison between Replication and Caching.

Caching systems	Replication systems
Pull-based (reactive) approach.	Push-based (proactive) approach.
Weak consistency model.	Strong consistency model.
Unreliable when the cache server is down, the requests will be redirected to the origin server.	Highly fault-tolerant due to object redundancy on other sites.
Low storage required.	High storage overhead.
Frequent cache replacement.	Long-term storage.
Low availability.	High object replica availability.
Does not need load balancing algorithms.	Requires efficient load balancing algorithms.
Traffic is reduced due to its reactive nature.	High traffic overhead unless efficient approaches are used.

1.4 Architecture of a WMN

The prevalent design for a WMN deployment is a two-tier architecture, wherein an access tier connects Mesh Clients (MCs) to stationary infrastructure nodes called Mesh Routers (MRs), and the MRs form a mesh wireless backhaul tier to route data packets between MCs in the WMN and between MCs and gateways that have an interface to the Internet [20, 21, 22, 23]. Currently, MRs are equipped with multiple radios that allow them to send and receive on multiple channels

in parallel increasing network capacity. There are many advantages of forming a WMN. For example, when enough neighbors cooperate to use their wireless home networks to forward each other traffic and form a WMN in a neighborhood. MCs do not need to individually install an Internet gateway, but instead they can share a fast, cost-effective access to the Internet via few gateways.

1.5 Research Motivation

Since the routing infrastructure in WMNs is open and modifiable, the mesh routers can be used both as replica servers (i.e., host) to increase the availability of content and as a relaying node. A cross-layer approach can be designed to improve the content placement and request routing schemes. This can be achieved by consulting the routing protocol augmented with MAC layer link quality metrics (i.e., ETX [24], ML [25], ETT [26]). In stationary WMNs, mesh routers normally have a low probability to leave the network and the high upload capacity in comparison with mesh clients, which motivated us to propose schemes for content replication at the mesh routers. The proposed content replication schemes take into account the variation of content popularity in the WMN over time.

The challenge of minimizing the access cost to the content is more serious in WMNs compared to the wired networks as the access cost for an object in multi-hop networks is identified by the number of hops between the requesting client and the nearest object replica server [7]. However, this cost is suitable for MANETs that usually use the hop count as a routing metric in which new paths must be found rapidly. However, high-quality routes may not be found. This is important in MANETs due to user mobility. In WMNs, the stationary topology can benefit from quality-aware routing metrics [27] that can be used to

augment cost minimization. The contention between neighboring mesh nodes for the wireless medium and the interference resulting from traffic on adjacent wireless links will result in a significant throughput reduction at peers when the packets traverse a long path in a WMN [28]. Minimizing object access cost results less bandwidth consumption in WMNs. Content replication is a widespread technique to improve the performance of object retrieval by incrementing the number of object replicas within the network subject to the limited storage capacity of a mesh router.

Related work was mainly focusing on content replica placement in the Internet, but when it comes to the wireless environment, most of the works are focused on caching and sharing content, whereas content replication and placement was not researched well in WMNs, which motivated us. Link quality metrics in WMNs capture the instant link quality, which helps in replica server selection. However, we cannot rely on using it in our cost function, since replica placement is intended for relatively longer periods. This has also motivated us to model the dynamic link cost that captures the long-term characteristics of a link.

1.6 Research Goals

Since mesh routers are scarce in resources (e.g., CPU and RAM), we consider in our design the cooperation among mesh routers to replicate and distribute contents via P2P communication in such a way to reduce latency and avoid congestion at the gateway(s); hence, our objectives are summarized as follows:

1. Increasing the availability of content, which reduces the download/lookup latency.

2. Find out what content to be replicated, the number of replicas needed in

the network, and the locations to place these replicas.

3. Find an efficient way for selecting the best replica server to serve clients' requests.

4. Alleviate the congestion problem at the gateway(s) by fetching the content from within the WMN, as a result, reduce the bandwidth consumption at the gateway(s).

1.7 Research Contributions

In this thesis, we first discuss the problem of content replication and placement in WMNs. We study the state of the art works related to content replication, especially in WMNs to have an image of what could be the problem in such context. We then model our problem as a Facility Location Problem. In particular, we find out that this is a variant of the p-median problem. Based on this finding, we propose and design two distributed schemes for this NP-complete problem to approximate the optimal solution according to our objective metrics. The two schemes exploit graph partitioning techniques to facilitate the distribution of the placement problem.

The schemes aim to minimize the object access cost leveraged by consulting the underlying routing protocol and metrics, at the same time considers the clients' demands and storage constraints in a best effort fashion, while preserving load-balance on participating nodes by adopting a P2P mechanism to distribute the replica role on mesh nodes so as to share the burden of content replication. The first scheme is hierarchical, whereas the second is flat. Based on the motivation and goals that we presented, our contribution in this thesis is summarized as follows:

- We formulate the replica placement problem as a p-median problem (Chapter 2) in a P2P approach, where a mesh router acts either as a server or as a client (on behalf of a mesh client).

- We survey the different approaches (Chapter 3) in content networking and classify them in the contexts of the Internet, MANETs and WMNs.

- We design a hierarchical, scalable, efficient, and distributed object replication and placement scheme (Chapter 4). The scheme partitions the network graph into different potential ones depending on the potential number of replicas per an object. The scheme builds a balanced binary tree composed of delegate nodes, where each one is responsible for a single replica placement. This converts the p-median problem into 1-median problem.

- We design a scalable, efficient, and distributed object replication and placement scheme (Chapter 5). This approach is different since it is flat (instead of the hierarchical one) and generates equal-sized partitions, where each partition is assigned a delegate node to divide the placement burden equally. In the scheme, each delegate node is responsible for a single replica placement, which converts the p-median problem into 1-median problem.

- We improve both of our schemes to consider the local popularity in a partition (Chapter 6). Two advantages can be achieved through this enhancement: (*i*) When an object is not feasible to place in a partition, then it will be forwarded to a larger partition, where a feasible placement can be considered; and (*ii*) It avoids congested links, therefore, the network performance improves.

- As MAC layer affects the link performance and consequently the route quality, we augment the placement decision (Chapter 7) by considering the

long-term link cost based on MAC layer link-quality metrics, whereas for server selection, a client's request is satisfied by the replica server that has the lowest (not necessarily the shortest) cost based on the instantaneous link-quality metric. We also improve both of the schemes by adopting a forecast model to estimate the content popularity for future replication periods.

1.8 Thesis Organization

The remaining of this thesis is organized as follows. Chapter 2 describes the network model considered and the system assumptions, then it gives a brief theoretical background for the Facility Location Problem and its variants, then we formulate our problem as a p-median problem. In Chapter 3, we present state of the art literature and discuss the shortcomings and/or the differences to our work. Chapter 4 describes and discusses the design of our first replication scheme (*SP-DNA*) and evaluates it using OMNeT++ simulator against other comparable schemes. In Chapter 5, we present another replication scheme (*MP-DNA*) and point out the differences with *SP-DNA*. Then we evaluate its performance using the simulation tool. We discuss the improvements on our schemes in Chapter 6 that consider the local popularity and its efficiency is evaluated against other schemes using the simulation tool. In Chapter 7, we build on the modified schemes and augment the placement with the link-quality metrics. We also show the benefit of using the forecast model we use and evaluate the performance. Finally, Chapter 8 concludes this thesis by summarizing the contributions of our study and outlining directions for future work.

Chapter 2

Background and Problem Formulation

The replication problem in WMNs requires that object replicas be stored at the smallest number of nodes as possible while the replica location satisfies the need to retrieve the content from interested users with minimum latency. With these properties, we find that the replication problem share many perspectives with the *Facility Location Problem* (FLP). Therefore, it can be casted as a FLP, which aims to minimize the average distance to access a facility. Up to our knowledge, the previous works on wireless networks consider caching as an approach to content networking, while others consider replica placement approximation algorithms for the wired Internet that use Linear Programming (LP) techniques. However, the drawback of these algorithms is that LP rounding usually involves solving large linear programs, which causes long running time [29].

In the following sections, we describe the network model (Section 2.1) that we consider in the context of this thesis, and then we give the reader a glimpse background (Section 2.2) on the Facility Location Problem, its variants, and why we use it to formulate the problem of object replica placement. Finally, we

formulate the placement problem (Section 2.3).

2.1 Network Model and Assumptions

In this section, we describe the network model of our object replica placement scheme over infrastructure/stationary WMNs. The system employs Mesh Routers (MRs) to act as content replica servers in a P2P fashion. We assume that MRs use IEEE 802.11 radios to build the wireless mesh backhaul infrastructure. MRs have replica server capabilities such as processing power and storage, in other words, a MR acts as a relaying node and as a replica server. Fig. 2.1 il-

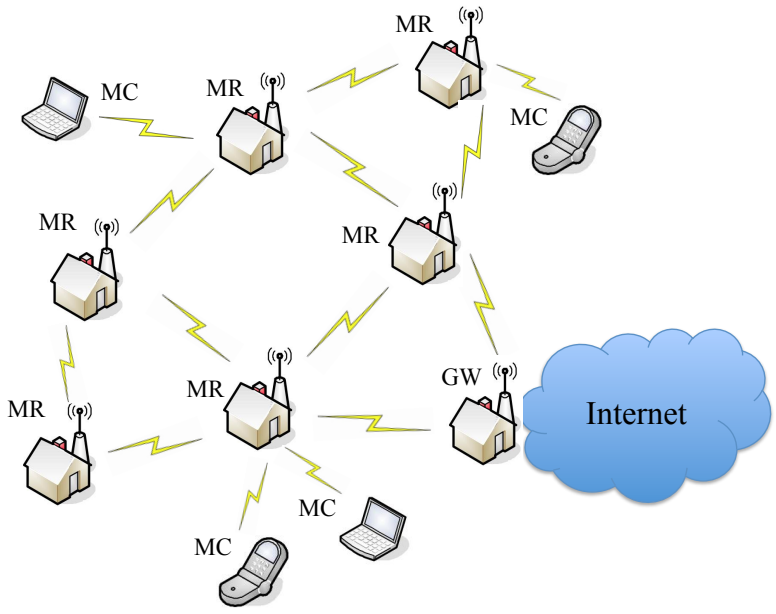


Figure 2.1: Network model considered in our proposed schemes.

lustrates the model considered in the thesis scope. It consists of mesh access points/MRs and Mesh Clients (MCs). The MRs are willing to participate in the replication system and are interconnected by wireless links to form a multi-hop backhaul infrastructure. One or more MRs are connected to the wired Internet

and are referred to as gateways (IGWs). MRs are used to access and relay packets, therefore, MRs support two types of interfaces for the wireless medium. The access interface offers network access for MCs, while the relay interface is used to relay client's traffic to its destination. Typically, the two interfaces work on non-overlapping channels to prevent interference with each other.

A mesh client (e.g., laptop or smart phone) is assumed to be either static or low in mobility, since the network scenario considered is a community WMN, where MCs are mostly home users that are associated with a nearby MR to access the mesh network. We also assume that a MC does not participate in packet relaying. To fetch an object (e.g., video file for a video on demand service), one of the structured P2P network directory services is employed (e.g., Chord [17]). As per the original Chord proposal, one of its applications is Chord-based DNS. DNS provides a lookup service, with host names as keys and IP addresses as values. This tells us that Chord does not necessarily impose the placement used by Chord. An application may call the function put(key,value) to store "value" with name "key", or get(key) to retrieve "value". For load balancing and fault tolerance, data items are often replicated at nodes other than the owner. In our case, since our schemes are responsible for the placement, the term value here is a pointer (IP address) to the hosting node and not the object itself.

A requested object is fetched first from within the participating MRs. Upon a fail, the request will be forwarded to the mesh backhaul to one of the IGWs, and then to the origin server. We also assume that the MRs are aware of the network topology and employ one of the widely deployed routing protocols such as OLSR [30] to find the routes with other MRs and to the IGW. OLSR is defined by the IEEE 802.11s standard as one of the future routing protocol implementations [31]. It also scales well for few hundreds of nodes. The scalability of OLSR is regarded to the Controlled-flooding approach used for exchanging

the Topology Control (TC) messages, which reduces control overhead. Flooding the network with routing updates may produce scalability issues,especially if frequent changes on medium conditions are considered. We assume that the MRs use TCP at the transport layer since it is broadly deployed for Internet access, and we assume that the exchanged messages between nodes are delivered in a reliable fashion and follow their transmission order. Our scheme is implemented over the underlying TCP protocol by a user space *ReplicaDaemon*. Finally, we assume that a data consistency model is used to insure that object replicas are exactly the same as the object in the origin server, moreover, objects are in Read-Only mode.

2.2 Facility Location Problem and Variants

Content placement in WMNs can be addressed by considering it as a *Facility Location Problem*. In operations research, optimizing the access cost of clients in different demand locations I to access a set of facilities J is an important problem. To address the problem, we need an efficient solution of a feasible cost (i.e., build and operate the facilities). This problem is formulated as a FLP such that many facilities are set up and each one is assigned with the demands of a subset of clients. There are several examples of FLP such as emergency points, education centers, public transport stations, and retail services.

We model the problem of content placement from the perspective of the FLP in the context of WMNs. In our model, a node can be either a demand node (client) or a facility node (replica server). FLP provides a mathematical formulation of optimization aspects. The formulation includes the cost to open facilities, and the distances between the clients and facilities that normally satisfy metric properties (e.g., the triangular inequality). The total cost relies importantly

on the number of facilities to open and their locations. To address the FLP, we need to specify the set of facilities to be opened, and assign the clients to the open facilities. For a typical FLP, J and I are the inputs. The output solution is to open a subset of the facilities $J \subseteq I$ and assign clients to their closest open facility. The solution tries to minimize the total cost that consists of two parts:

1. **Opening cost**: The opening cost f_j for a facility j depends on the targeted problem. For a solution, the opening cost is the sum of costs for all open facilities.

2. **Weighted distance cost**: This is the weighted distance cost from a client $i \in I$ to a facility $j \in J$ denoted by $h_i d_{ij}$. Where, h_i is the demand at node i and d_{ij} is the shortest distance from node i to facility j. It is assumed to be symmetric and satisfies the triangle inequality. The sum of distance costs of all clients is the total cost for the solution.

A number of variants for the FLP exist by combining these costs in different ways [32, 33, 34]. The number of facilities to open can be a constant (p-median problem), limited number of served clients by a facility (capacitated FLP) or unlimited (uncapacitated FLP). Most variants of FLP are NP-complete [35]; hence, approximation algorithms that find solutions close to the optimal solution are under investigation. Following are some variants of the FLP:

2.2.1 *p*-center problem

This is also known as the *minimax problem*, as we seek to minimize the maximum distance between any demand and its nearest facility. The cost here is not weighted by h_i. Then $\forall j \in J$, select up to p facilities to minimize the total cost:

$$C(I, J, p) = \sum_{\forall i \in I} \sum_{\forall j \in J} d_{i,m(i)} \tag{2.1}$$

where $m(i) \in J$ is the facility j closest to i.

2.2.2 *p*-median problem

When a limited budget is available for opening the facilities, and the opening cost of all the facilities are approximately the same. Then $\forall j \in J$, select up to p facilities to minimize the total cost:

$$C(I, J, p) = \sum_{\forall i \in I} \sum_{\forall j \in J} h_i d_{i,m(i)} \tag{2.2}$$

where $m(i) \in J$ is the facility j closest to i.

2.2.3 Uncapacitated FLP (UFLP)

When the opening cost is considered and the number of facilities to open depends on a joint optimization for opening cost and distance cost, we have the UFLP. The solution is to open a set of facilities J to minimize the joint cost $C(J, I, f)$, where a facility j can serve an unlimited number of clients:

$$C(I, J, f) = \sum_{\forall j \in J} f_j + \sum_{\forall i \in I} \sum_{\forall j \in J} h_i d_{i,m(i)} \tag{2.3}$$

where $m(i) \in J$ is the facility j closest to i.

2.2.4 Capacitated FLP (CFLP)

In CFLP, we assume that a facility can have a constraint in resources dedicated to its clients, so it is important to limit the number of clients assigned to a facility. We open a set of facilities J to minimize the joint cost $C(J, I, f)$, while ensuring that each facility j can only serve at most u_j clients:

$$C(I, J, f) = \sum_{\forall j \in J} f_j + \sum_{\forall i \in I} \sum_{\forall j \in J} h_i d_{i,m(i)} \tag{2.4}$$

where $m(i) \in J$ is the facility j closest to i and c_j is the number of clients i attached to facility j such that $c_j \leq u_j$.

2.2.5 Multiple commodity facility location problem

The FLP can be extended to a problem of multiple commodities served at a facility. Let L denote the set of commodities $L = 1, ..., M$. Each commodity $x \in L$ has a subset of clients. To extend the cost function, we consider an optimization for all commodities and assume the same opening cost f for every commodity x, the joint cost can be expressed as:

$$C(I, J, L, f) = \sum_{\forall j \in J} \sum_{\forall x \in L} f_{j(x)} + \sum_{\forall i \in I} \sum_{\forall j \in J} \sum_{\forall x \in L} h_i d_{i,m(i,x)} \tag{2.5}$$

where $m(i, x) \in J$ is the facility j holding x closest to i. If we consider the CFLP, we have the number of clients i demanding any commodity x attached to facility j such that $\sum_{\forall x \in L} c_j(x) \leq u_j$.

2.3 Problem Formulation

In this section, we introduce the problem formulation and the optimization goal in the context of the network model. The notation given in Table. 2.1 is used in the problem formulation. We view the WMN as an undirected, connected graph, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, .., n = |\mathcal{N}|\}$ is the set of graph nodes, and $\mathcal{E}$ is the set of edges. Each edge $e \in \mathcal{E}$ is assigned an arbitrary, non-negative weight. The distance between two nodes i and j is the sum of the weights of edges along a shortest path between i and j. This distance may reflect several metrics such as the number of traversed nodes (hops), latency, link-quality routing metrics (i.e., ETX, ETT, ML), servers' load, etc. We formulate this problem as a p-median problem [32] for the following reasons: (*i*) Since the opening cost in our model

Table 2.1: Notation used in the problem formulation.

Symbol	Meaning
$\mathcal{N}$	Set of demand nodes indexed by i, and the set of potential facility locations (replica servers) indexed by j.
$\mathcal{M}$	Set of distinct objects within the WMN indexed by m.
$\mathcal{O}_m$	Identifier of object m.
$\|\mathcal{O}_m\|$	Size of object m in bytes.
$\mathcal{P}r_i(m)$	Popularity (demand) of $\mathcal{O}_m$ at node i.
p_m	Number of facilities (replicas) of object $\mathcal{O}_m$ to establish (locate).
d_{ij}	Distance between demand node i and potential serving node j that may reflect several metrics such as hop-count, latency, link quality routing metrics (e.g., ETX or ETT).
S_i	Storage capacity at node i.
x_j	A binary decision variable = $\begin{cases} 1 & \text{if we locate at site } j, \\ 0 & \text{otherwise.} \end{cases}$
y_{ij}	A binary decision variable = $\begin{cases} 1 & \text{if demand node } i \text{ is assigned to a facility at node } j \\ 0 & \text{otherwise.} \end{cases}$
x_{im}	A binary decision variable = $\begin{cases} 1 & \text{if object } m \text{ is stored at node } i, \\ 0 & \text{otherwise.} \end{cases}$

represents the migration of objects between replica servers for the new placement to be established, and because changing the placement is performed periodically, the *opening cost* $\ll$ *weighted distance cost*. This cost of object migration –in some cases, objects remain hosted for multiple replication periods if they remain popular– is not comparable with the cost of serving clients requests unless the replication period is very short, which is not the case because replication has a computation/communication cost that requires the replication period to be meaningful; and (*ii*) We consider the demand-weighted distance as an objective to be minimized. The p-median problem can be formulated as follows:

$$Min \sum_{i,j \in \mathcal{N}} Pr_i(m) d_{i,min(i,m)} y_{ij} \tag{2.6}$$

Subject to:

$$\sum_{j \in \mathcal{N}} x_j = p_m \tag{2.7}$$

$$\sum_{j \in \mathcal{N}} y_{ij} = 1 \quad \forall i \in \mathcal{N} \tag{2.8}$$

$$y_{ij} \leq x_j \quad \forall i, j \in \mathcal{N} \tag{2.9}$$

$$x_j \in \{0, 1\} \quad \forall j \in \mathcal{N} \tag{2.10}$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j \in \mathcal{N} \tag{2.11}$$

$$\sum_{m \in \mathcal{M}} |\mathcal{O}_m| x_{im} \leq S_i \quad \forall i \in \mathcal{N} \tag{2.12}$$

The objective function (2.6) minimizes the demand-weighted total distance traveled, while $min(i, m)$ is the facility j holding $\mathcal{O}_m$ closest to i. Constraint set (2.7) stipulates p_m facilities to be located $\forall m \in \mathcal{M}$. Constraint set (2.8) requires each demand node to be assigned to exactly one facility. Constraint set (2.9) restricts demand node assignments only to open facilities. Constraint set (2.10) establishes the siting decision variable as binary. Constraint set (2.11) stipulates the

demand at a node to be assigned to one facility only. Constraint set (2.12) requires that the storage is constrained by the space available at node i. Our goal is to find a placement for every $\mathcal{O}_m$ such that it minimizes the demand-weighted total distance. The p-median problem simply states that: Given a graph $\mathcal{G}$, find $\mathcal{V}_p \subseteq V$ such that $|\mathcal{V}_p| = p$, where p may either be variable or fixed, and that the sum of the shortest distances from the vertices in $\{V \backslash \mathcal{V}_p\}$ to their nearest vertex in $\mathcal{V}_p$ is minimized.

When applied to a general network, the p-median problem can be difficult to solve to optimality (this class of problems is NP-complete). Limiting potential facility locations to network nodes, however, reduces the number of possible location configurations to

$$\binom{\mathcal{N}}{p} = \frac{\mathcal{N}!}{p!(\mathcal{N}-p)!} \tag{2.13}$$

where $\mathcal{N}$ represents the number of nodes in the network. Thus, for a fixed value of p, the p-median problem can be solved in polynomial time. Nevertheless, a total enumeration approach would be computationally prohibitive for reasonable values of $\mathcal{N}$ (hundreds to thousands of nodes) and p (tens of locations sited). For variable p, the problem is NP-Complete [36]. Such complexity issues have led to the development of sophisticated algorithms for solving this problem.

The formulation presented above suggests the use of integer programming techniques for solving p-median problems. While these techniques are often able to reach integer optimal solutions for moderately sized problems in a reasonable time. However, when the problem size is large (in terms of p and $\mathcal{N}$) efficient heuristics are needed to solve the problem.

2.4 Summary

In this chapter, we presented the network model considered throughout the thesis and the assumptions in our system. Then, a general overview of the facility location problem was given along with its variants to show how the replica placement problem can be modeled. Inside the scope of the facility location problem, we started by introducing the problem formulation as a p-median problem and discussed why we chose this model. We also discussed the objective function and the constraint set.

Chapter 3

Literature Review

The new era of Web applications do not only provide access operations on content. Moreover, creating and modifying content and placing content in feasible locations. To deal with the aforementioned requirements, new forms of CDNs are introduced; hence, content distribution and management is bringing new challenges in this domain. Although WMNs have advantages like easy deployment, low infrastructure cost, self-organization, and redundant multi-paths. However, content delivery in multi-hop WMNs faces many challenges such as limited and dynamic bandwidth along the path, interference resulting from shared medium, the effect of multiple relay nodes on the throughput, user mobility, and most of all is the link fluctuation that could be resulting from channel fading, external interference, and weather changes. All these factors lead to a dynamically changing topology, although the nodes are stationary.

A WMN –compared with the traditional wired Internet– suffers (relatively) from fluctuation in the path over time, the wireless link quality that relies on the signal strength, the data rate per link, packet loss, congestion control, and contention between nodes. Considering all these factors could bring down the throughput, which affects Quality of Experience. Previous works on content

delivery are based on overlay networks, where the underlying layers are transparent (i.e., the MAC, PHY and Network layers). However, in WMNs, it is very important to consider these layers as they play a crucial role in the whole network performance, and we can minimize the communication cost for a requested content.

Up to our knowledge, the previous works consider caching as an approach to content networking, while others consider replica placement approximation algorithms for the wired Internet that use Linear Programming (LP) techniques. However, the drawback of these algorithms is that LP rounding usually involves solving large linear programs, which causes long running time [29].

Many Replica Placement Algorithms (RPAs) have been proposed for the Internet and MANETs, but WMNs have received much less attention in this area. On one hand, RPAs designed for the Internet are centralized and incur a high computation cost. On the other hand, replication schemes designed for MANETs focus on issues such as low bandwidth and energy constraints. In this chapter, we present the major studies for caching and replication techniques in the scenarios of the wired Internet, Wireless Mesh Networks, and Mobile Ad Hoc Networks.

3.1 Replication Schemes in WMNs

MeshChord was proposed in [37], which is a modification to the Chord [17] protocol that is used for P2P DHTs. The idea is to make use of locality by assigning peers, which are close in the physical network with close-by IDs in the Chord ring. Since in Chord, most of the messages are exchanged between a peer and its successor/predecessor, peers in the same sub-region of the deployment area are mapped to the same segment of the Chord ring, which converts physical proxim-

ity into proximity in the Chord ring. The location-aware ID assignment requires that peers are aware of their location, which can be done using GPS receivers. The second contribution of MeshChord is a MAC cross-layering technique that aims at speeding up the lookup operations by exploiting the information that is available at the MAC layer due to the 1-hop broadcast communication occurring in wireless networks. A peer may be able to resolve a lookup request that is physically close to the peer issuing the lookup, while they are far away in the Chord ring. A peer receives a packet at the MAC layer, and then sends it up to the application layer for further processing. If the packet was not destined to it, then it checks if it may resolve the lookup operation. In this case, it sends a message containing its own ID to the peer that invoked the lookup, which may accelerate the lookup operation.

In [38], P2PMesh was proposed. It aims to reduce both the number of failed lookups and the file lookup latency. It operates as follows: When a peer has a file to share, it requests to upload the file to the mesh router it is connected to, and after acknowledgement, the peer uploads the file to the mesh router, which in turn registers the file's descriptor, ⟨key, MR's IP address⟩. When a peer requests a file, it hashes the file's metadata to obtain the key, which is sent to the peer's current mesh router. The overlay P2PMesh tries to locate it within the mesh network. If the requested file is not in the peer's mesh router, the key is routed to the mesh router responsible for that key using any DHT routing protocol, if it is not found in the mesh network, the key is sent to the gateway to obtain the file from the Internet. When the P2PMesh system locates the requested file at one or more nodes, it returns a list of sources to the requester, which selects this set of file providers to retrieve different chunks from. The protocol selects file providers based on minimizing the following routing metrics: (*i*) Route coupling that results in interference between neighboring routes destined to the

same receiver; (*ii*) Hop distance between requesting node and file provider; and (*iii*) Number of disjoint nodes on a route from the requester to the provider. We note here that content placement was not considered.

The work in [39], proposes a cooperative file transfer protocol, where a file is split into chunks of fixed size, which are numbered in ascending order from the beginning to the end of the file. The requester performs a peer discovery procedure to find potential download peers sharing chunks of the file. Then, the requester runs a peer selection algorithm to determine a set of active download peers. The rarest chunks are downloaded first using the TCP protocol by explicitly requesting the chunk number. The protocol monitors the performance of a download using end-to-end goodput measurements. For each current file download, the protocol maintains a data structure denoted as File Distribution Table (FDT), where each entry contains the IP address of a download peer and the numbers of chunks it possesses. The following two phases are performed to lookup a file:

1. **Peer discovery**: A QUERY message with the file ID and the chunk numbers is composed and flooded to all reachable nodes. If a peer possesses the requested chunks, it replies with a RESPONSE message with the file ID and the available chunk numbers via unicast. The requester updates its FDT. The QUERY/RESPONSE messages are piggybacked on RREQ/RREP messages of the routing protocol to reduce control traffic.

2. **Peer selection**: Peer selection is not based on the number of hops since the impact of background traffic on TCP goodput shows that the shortest path does not always provide the best goodput.

Fig. 3.1 depicts the effect of background traffic on goodput. The interference on the short path increases due to the traffic of FTP 1, which decreases the achieved

goodput significantly. When FTP 3 is out of the interference range of FTP 1, it achieves around 66% more goodput than FTP 2, which runs on the short path. Hence, the protocol selects download peers with the least-loaded paths.

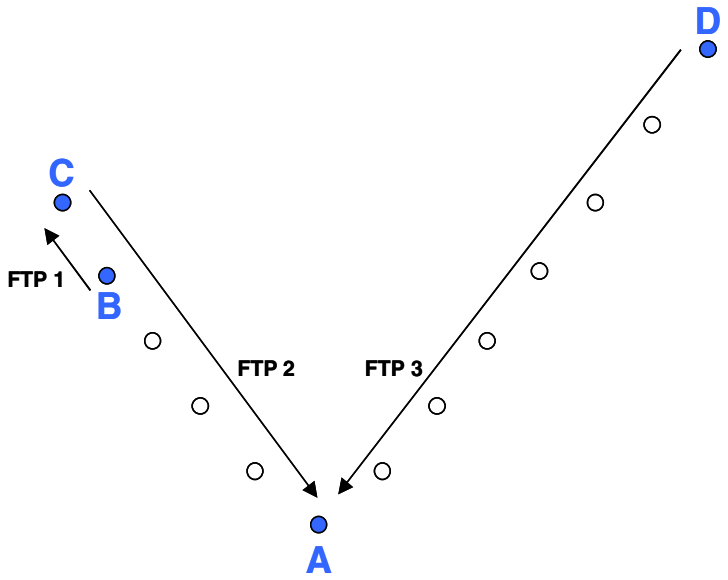


Figure 3.1: Effect of background traffic.

In [40], the authors propose two policies to assign clips to peers in a WMN, Frequency-based and Byte-hit policies. Both of them sort the clips based on the popularity of each clip with the objective to place the highest popular clips that can be accommodated locally in each mesh node's storage. To differentiate between the two policies, the Frequency-based does not consider the clip size in the decision of selecting the clips to be replicated, while the Byte-hit policy divides the clip's popularity by the clip's size, then it sorts the clips and selects the top clips that can be accommodated locally in each node. They found that Byte-hit is superior to Frequency-based for two reasons. First, it maximizes the number of peers that can simultaneously display their referenced clips. Second, it is more robust to error in access frequencies. However, we argue here that this will lead to placement of clips on nodes that do not show popularity for the

clip(s), which leaves the local demands unsatisfied efficiently.

H2-VIP replication scheme was proposed in [41], which can minimize the system failure rate by allocating a small amount of extra space. The main benefit of H2-VIP is that it can compute the optimal number of replicas of video blocks, such that the overall system failure rate can be minimized subject to a limited storage space. The spare space of each home device can collectively form a large community storage farm. In H2-VIP, the following steps are carried out:

1. Each video clip is divided into equal-sized blocks.
2. The first block of each video is stored on all nodes.
3. For each remaining block, computes its delay tolerance.
4. According to the delay tolerance, computes the number of replicas of a block.
5. Compute the extra number of replicas of video block, according to the video access probability, and the total extra storage space, allocated to improve the system reliability.
6. Finally, all blocks are scattered over all peers (or home devices) according to the delay tolerance.

The H2-VIP scheme improves the robustness against failure (i.e., a peer home device goes offline). However, there are two differences in our scenario, the first is that we consider the content placement in the mesh nodes rather than in the home devices' storage; hence, the reliability is higher. Although a mesh node might go offline, but it does not happen frequently as home devices do. The second difference is that we aim to disseminate the content efficiently based on clients demanding it and that was not considered in H2-VIP.

In [42], the authors proposed a suboptimal solution for caching in multi-hop wireless networks by caching data at a predefined set of nodes across the network. They design caching strategies that optimally trade-off between overhead cost and access latency. The caching strategies improve the system performance in terms of throughput and access latency. However, there is no peer cooperation between the mesh nodes, since the cache servers are fixed and limited to a subset of the nodes. They formulate the problem as a special case of the connected facility location problem, called the rent-or-buy problem that aims to select the sites (caches) to build the facilities and connect them by a Steiner Tree [43] to minimize the access cost. However, this model forces the set of caches to be connected in the Steiner Tree, and they assume equal service demand (popularity). These two characteristics do not apply to our scheme since replicas do not need to be connected and different nodes have different demands. The work in [44], proposes an incentive scheme for multimedia services in 3G/WLAN dual mode network in which two types of users are assumed. Premium users who are willing to pay for the contents and ordinary users that do not. The premium users receive high quality contents from the provider at a discounted price and share it with ordinary users. The authors in [45], show that multi-hop communication is a sustainable scheme for certain values of file popularity, cache, and network size. They formulate the joint problem of replication and routing and compute an order optimal solution. They propose information theoretic scaling regimes to compute the required link capacity for a static cache placement in a multihop wireless network.

3.2 Replication Schemes in MANETs

In [46], the main idea of the Replica Distribution (RD) protocol is to disseminate object replicas on nodes at r-hop distance, where the value of r-hop depends on the replication degree of each object that is found. When the node joins the network, it communicates the description of its objects to the manager that decides the replication degree of each object on the basis of the provided descriptor and the estimated number of nodes in the dense region. When a node needs to replicate an object, it sends a replication packet specifying the number of replicas remaining to replicate and the desired r-hop distance between replicas. The replication packet propagates on nodes along an approximately straight line with a fixed direction. When the packet reaches the r-hop away node, it saves a copy and reiterates the process by decreasing the number of requested replicas. However, this placement does not consider object popularity in the placement decision, which leads to inefficient dissemination for the content.

The work in [47] proposed a self-stabilizing asynchronous, fully distributed, scalable protocol that places replicated resources in a network of arbitrary topology with the aim that the furthest distance to be traveled to find a content replica is slightly larger than optimal, and the distance between identical copies is large. They modeled the problem as a p-center problem, where the objective function is to locate p facilities such that the maximum distance is minimized.

$$Min \sum_{x} \sum_{i} d_t(x, c_i) \tag{3.1}$$

where $d_t(x, c_i)$ is the distance at time t from node x to center c_i. The objective is to place different items in the vicinity of each node or to place the identical items as far away from each other as possible. This is similar to the p-center problem formulation and can be used for channel assignment to maximize channel reuse. They found the protocol is close to optimal in convergence time

and message overhead. However, in WMNs, the requirements are different for replication strategies since they are less constrained by energy consumption and node mobility. Furthermore, the p-center problem does not consider content popularity at each node. Therefore, our model is different, since we consider popularity as a factor affecting the placement.

Random-Walk Diffusion (RWD) mechanism was proposed in [48] in which a mobile device hosting a content replica, stores it for a storage time t. At the end of its storage time, the replica node selects with equal probability one of its neighbors to store the content for the following storage period. Therefore, content replicas roam the network by moving from one node to another, randomly, at each time step t. In [49], the main contribution is the design of a mechanism for content placement and replication that achieves load balancing according to the variations in the network topology and the query rate. The mechanism distributes the burden of storing and providing content on nodes to achieve load balancing. The replica nodes are responsible to decide whether to replicate, hand over or drop content based on local measurements of their workload. During storage time τ, the replica node counts the number of queries that it serves (i.e., $S_{v(j)}$). When the storage time expires, the replica node compares $S_{v(j)}$ to a reference value S_R for the workload that node $v_{(j)}$ is willing to support. Decisions are taken as follows:

$$\text{if } (S_{v(j)} - S_R) \begin{cases} > \epsilon & \text{replicate,} \\ < -\epsilon & \text{drop,} \\ \text{else} & \text{handover} \end{cases} \tag{3.2}$$

Where ϵ is a tolerance value to avoid replication/drop decisions in case of small changes in the node's workload. In both [48] and [49], the mechanisms do not replicate the content according to the clients' demands, where the former mechanism guarantees the existence of one replica at anytime and the latter

mechanism replicate, handover or drop the replica.

In [50], the authors propose the use of social networking concepts to place shared data efficiently in an opportunistic network. They introduced the concept of *conditional betweenness centrality* that measures the cost of accessing content on a particular node from any other node that has interest in the content. Initially, content is placed at a random node, then the algorithm finds the shortest paths from all nodes to the content location. Some nodes might have multiple shortest paths passing through them. A portion of the top nodes is selected and for each one of them, the centrality value is computed and the node with the lowest cost of data access is selected to be the location of the data. This process is repeated until there is no more movement of the data. However, Content is not replicated over multiple nodes (1-median), while we emphasize that content replication decreases communication overhead and increases the availability in presence of node failures. The authors in [51] propose a P2P content sharing protocol for wireless ad hoc networks. They study the best neighborhood selection strategy that suites the wireless multi-hop environment by organizing peers in a minimum spanning tree and define the neighborhood of a peer as being its neighborhood over the logical tree rather than its physical neighborhood.

In [52], SCALAR (SCAlable data Lookup And Replication framework) was proposed for MANETs. The framework does not depend on the underlying routing protocol and minimizes the number of nodes involved in the data lookup process by constructing a dynamic virtual backbone structure among the mobile nodes. In [53], the REDMAN (REplication in Dense MANETs) middleware solution was proposed. The main idea is to maintain a fixed replication degree for the needed resources regardless of replica server nodes exiting the dense region. *Zone-Based Replication* (ZBR) scheme was proposed in [54] for MANETs, where an object is replicated if it is not within the zone of the requesting node.

The replica placement gives priority to peripheral nodes along the route to access the object, such that enough storage is available. However, our scheme is different as we consider the factor of object popularity in the placement decision. In [55], the main contribution is the design of a mechanism for content placement and replication that achieves load balancing according to the variations in the network topology and the query rate. The mechanism distributes the burden of storing and providing content on nodes to achieve load balancing. The replica nodes are responsible to decide whether to replicate, hand over or drop content based on local measurements of their workload.

3.3 Caching Schemes in MANETs

In [56], the mobile client has a cache storing the frequently accessed objects. The cached objects are assumed to satisfy the local requests of the client and the requests passing through it from other clients. If an object miss occurs in the local cache, the mobile client first searches the data in its zone (i.e., one-hop neighbors). On a miss, it forwards the request to the next client that is on the path towards the data center, which searches its local cache and its zone's cache. The process is repeated until reaching the data center. The algorithm allows neighboring mobile nodes to share their data that helps reduce both the average query latency and the limitations on data access. However, the latency perceived might be longer than forwarding the request to the data center, especially if the neighbors of intermediate nodes do not have a cached copy of the requested object. Another shortcoming is that there is no cooperation between caches beyond each one's zone (i.e., other than the data center direction).

In [57], the concept of Neighbor Caching (NC) is proposed, where the basic idea is to utilize the cache space of inactive neighbors for caching tasks. When

a node accesses an object from a remote node, it caches the object in its own cache space, if this operation requires evicting an object from its cache (based on a replacement policy) then the evicted object is stored in the idle neighbor node's storage. For future requests of the object that migrated to the neighbor node's cache, it can be served from the neighbor node instead of the far remote source node. The algorithm utilizes the cache space of inactive neighbors. However, this approach does not utilize an efficient cooperative caching between nodes. Furthermore, because of the dynamic nature of caching and mobility of nodes, both the miss ratio and the access latency increase. The authors in [58] present an architecture for content-centric MANETs called CHANET (Content centric fasHion mANET) that copes the highly dynamic topologies in MANETs. CHANET is built on a connectionless content-centric layer on top of the IEEE 802.11 protocols. It exploits the broadcast Interest and Data packets such that the receiving node takes a local forwarding decision based on packet overhearing. In [59], SPontaneous Information and Resource sharing InfrasTructure (SPIRIT) was proposed to allow mobile devices to create, discover, join, leave, and revoke the sharing of resources in an efficient and robust fashion. SPIRIT is built on top of a group communications layer for mobile devices that allows users to express heterogeneous services and service sharing paradigms using an ontology-based subscription language.

In [60], the authors propose a cooperative caching scheme GroupCaching for MANETs, where the mobile nodes maintain the localized caching status among the group members; hence, they can cooperate to store different objects. Furthermore, if a node has free cache space, then its neighbors can utilize it. Each node and its one-hop neighbors form a group. A node has a group member ID and sends its caching status to its group members. Each node maintains two tables self_table and group_table. The Placement and Replacement Policy works

as follows: When a node receives an object from the destination, it caches the object if it has enough space. Otherwise, it checks the available spaces in other members, if there is sufficient space, it places the object randomly in one of the members cache. Otherwise, it looks up the group_table to see if the object already exists. If yes, the object is not cached. Otherwise, it selects the member that has the oldest timestamp of the cached object and sends the object to that member for replacement. When the member receives the object, it repeatedly performs the LRU replacement to increase available space until the received object can be cached. The scheme improves cooperation between mobile hosts to store different data objects. It also utilizes the available cache space of neighboring mobile hosts. However, redundant caching increases, since the object will be cached locally without insuring that another copy may exist in the one-hop neighbors' caches. Furthermore, cache discovery involves the group caches only, but does not query neighboring groups' caches. On a group cache miss, the query is forwarded to the data source, which makes it a bottleneck.

COOP was proposed in [61], which is a novel cooperative caching service for data access applications in MANETs. The service introduced three basic cache resolution schemes:

1. **Adaptive flooding**: Flooding helps discovering the closest cache around the requester, while in the meantime; it helps the neighbor nodes learn from this announcement who has the data next time it is requested.

2. **Profile-based resolution**: Previously received requests are recorded in a table called Recent Requests Table (RRT). A node checks the RRT after its local cache misses and before flooding a request. If an entry is found, the node compares its distances to these matching caches and the original data source, and selects the closest one. Otherwise, adaptive flooding is used.

3. **Roadside resolution**: If no cache is found using previous schemes, the data request is forwarded to the original data source, which allows a node on the forwarding path to serve as a proxy to resolve the request.

These schemes improve data availability and access efficiency. However, Flooding incurs heavy traffic overhead for content discovery. Moreover, the RRT caches can add more lookup latency as cache contents change rapidly. Moreover, the Roadside Resolution is the basic scheme that is used in traditional Web proxies.

In [62], the authors propose the COACS system (Cooperative and Adaptive Caching System). The system minimizes delay and maximizes the likelihood of finding data that is cached in the ad hoc network, without adding excessive large traffic at the nodes. COACS is a distributed caching system that aims to index cached queries for efficiently and reliably finding the cached objects. Nodes can play one of two roles: Caching Node (CN) and Query Directory (QD). The QD is meant to cache queries issued by the requesting node, while the CN's task is to cache objects (query responses). When a node requests a content that is not cached (a miss), the database is accessed to locate a nearest copy. When the response is received, the Requesting Node (RN) will act as a CN and caches the object. The nearest QD to the CN will cache the query and make an entry in its hash table to link the query to its response. QDs act as distributed indexes for previous requests and responses by storing entries for the requests and the corresponding CNs addresses storing the responses. COACS improves the hit ratio and reduces access delay but at the cost of a higher bandwidth consumption introduced by message overhead. However, the query request flows from one QD to another that may increase the cache discovery latency. Moreover, there is only one QD that knows about the cached object in the corresponding CN. Asymmetric caching was proposed in [63] as

an overlay solution on the baseline network deduplication that allows the *dedup destination* to selectively feedback appropriate portions of its cache to the *dedup source* aiming to improve the redundancy elimination efficiency. They show that the scheme can identify and eliminate a significant amount of redundancy.

The work in [64] proposed three cooperative caching techniques to augment data access in ad hoc networks. The first scheme is the CacheData scheme, where a node caches a passing-by object when it finds that it is popular. Fig. 3.2 (reprinted from the authors' paper) can help illustrating how the techniques work. For example if nodes 6 and 7 request an object through node 5, then node 5 would know that the requested object is popular and caches it. Future requests by nodes 3, 4 and 5 can be served by node 5. A conservative rule should be followed: A node does not cache the data if all requests for the data are from the same node. The second scheme is CachePath, where a node caches the path of

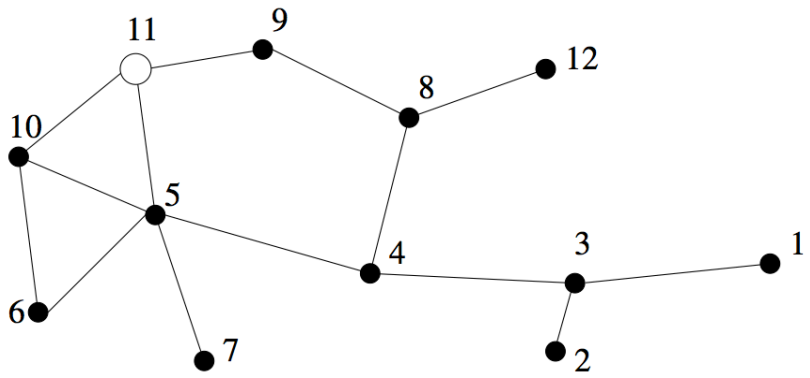


Figure 3.2: CacheData and CachePath.

previously served objects. For instance, if node 1 requested an object from node 11, then node 3 remembers that the object is cached in node 1 and in the future, if node 2 requests the same object from node 11, then node 3 will redirect the request to node 1. They propose an optimization method that is: A node need not cache the path if it is closer to the data center (node 11) than the caching

node. The third scheme is called HybridCache, which takes the best of both of CacheData and CachePath. Based on a specified criteria (objects' size, TTL and the Hsave) a node caches the data or path, where each criterion has a specified threshold:

1. If size is small, CacheData is favored as it takes less space in the cache.
2. If TTL is small, CachePath is not preferred as the object will soon be invalid.
3. If Hsave is large, CachePath is chosen because it can save a large number of hops.

However, the schemes have the following shortcomings:

1. In CachePath, since the nodes are mobile, redirecting a request to an out of range node represents an overhead on the redirecting node, which will then resend the request to the data center increasing the access latency.
2. In CacheData, the data may be replaced due to the cache size limitation and therefore, a lot of faulty redirections might be incurred, especially when the content is frequently being replaced. As a result it cannot be reliable and its performance decreases dramatically.
3. We also note that there is no full cooperation between caching nodes as each node performs the caching tasks independently. The nodes do not offer functionality for searching the contents of all other nodes.

In [65], the authors proposed an asymmetric cooperative cache approach, where object requests are transmitted to the cache layer on every node, but the data replies are only transmitted to the cache layer at the intermediate nodes that need to cache the data. This reduces the overhead of copying data between the user space and the kernel space. As depicted in Fig. 3.3 (reprinted from the

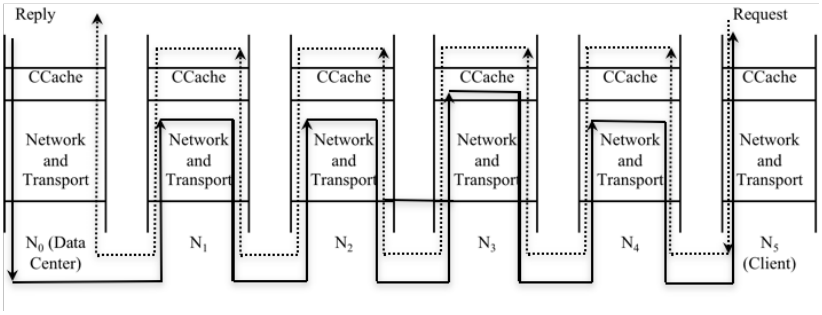


Figure 3.3: Asymmetric cache request/cache reply.

authors' paper), data requests and data replies are treated asymmetrically. A request message follows the path passing through the cache layer in every node in the path, while the reply message follows a different path. If none of the nodes in the path need to cache the returned object from N_0, then N_0 sends the object directly to N_5. However, if N_3 needs to cache the object, when N_3 receives the request message, it modifies the message and notifies N_0 that the object should be sent to N_3. The replied object will visit the cache layer only in the intermediate nodes that have shown an interest in the object (i.e., N_3). The benefit that is achieved is minimizing the object copying overhead between kernel and user spaces. However, the content lookup considers only the nodes on the route from the client towards the data center, but it does not consider neighboring nodes that could be closer than the data center. Both of the works in [66, 67] are based on the Content-Oriented Publish/Subscribe System (COPSS). In [66], the authors proposed a hybrid content centric architecture based on publish/subscribe systems that are built on top of Content Centric Networks (CCN). The proposed hybrid-COPSS aims to address both the need for incremental deployment of CCN, and combines the functionality of CCNs with the efficiency of IP forwarding by integrating IP multicast to achieve forwarding efficiency using shortest path routing. While in [67], they propose Gaming over COPSS (G-COPSS), which

is a distributed communication infrastructure using COPSS to enable efficient decentralized information dissemination in Massively Multiplayer Online Role Playing Games (MMORPG), jointly exploiting the network and end-systems for player management and information dissemination. The aim of G-COPSS is to scale well in the number of players in a single game, while still meeting users' response time requirements.

3.4 Caching Schemes in WMNs

Ditto was proposed in [68], which is a system for opportunistic caching in multi-hop WMNs. It caches data either when it sends it to a client, or when it overhears it being transferred by other nodes. It divides content into chunks of data that can be cached in the nodes along the data path and in those overhearing the wireless transmissions. Ditto works much like hierarchical web caching, where each Ditto proxy serves the data to its previous hop, either from its cache or by requesting it from its next-hop Ditto proxy. Each proxy caches all chunks passing by and/or overheard chunks. To locate chunks as many nodes in the mesh might have cached redundant chunks, the next-hop proxy up to the gateway is chosen forming a tree rooted at the gateway. They design the Sniffer module, where it reassembles chunks when hearing different parts of the chunk from different TCP streams. This scheme favors other schemes that only cache along the actual data transfer path. It also increases the hit ratio by caching overheard objects. However, it lacks cooperation between caches as on cache miss, the content is looked up along the path to the data source. Another shortcoming is that, it does not eliminate redundant cached copies leading to inefficient utilization of the caches' storage capacity.

In [69], the focus was to determine the optimal number of replicas to minimize

object access cost (defined as the Euclidean distance from the requester to the nearest replica) in multi-hop wireless mesh network, when the prior knowledge on the global popularity of the objects is available. They also proposed a local replacement algorithm to approximate the optimal strategy without any prior knowledge on the object popularity. However, it ignores the control message transferring cost (use flooding to discover replica servers). Moreover, the replacement decision is taken locally, which might lead to unbalanced placement within the neighborhood, resulting in-efficient utilization of storage. Moreover, it does not consider other cost factors (i.e., link quality). They modify the *GreedyDual* algorithm [70] for cache replacement to approximate the optimal strategy.

UPAC (Unified P2P and Cache) system was proposed in [71], which employs multiple mesh content cache servers and P2P technique. Here, the mesh router plays two roles, VoD streaming server and peer. Two scheduling schemes are used, streaming and downloading. Client devices, if available in the mesh, also serve as best effort peers to further reduce the network resource consumption along the path from the source to the gateway, and balance the network traffic load. A client device can form a P2P relationship with mesh content servers and other peer devices. Meanwhile, it establishes a client-server relationship with mesh content servers. To reduce the load on the servers, the requested content is looked-up first at one of the neighboring clients and in case of a hit; the content is forwarded in a P2P fashion. Otherwise, it is looked-up in cache servers for availability and the most suitable primary and secondary servers are selected to serve the request. They suggest several schemes for server discovery and selection:

1. **Centralized scheme with server load as the selection metric**: A client sends a request to the main server, which selects a primary and secondary mesh servers, then it informs the client of both servers. The selection is

based on the least load. However, this requires the mesh servers to report their loads to the main server periodically.

2. **Overlay scheme with end-to-end delay as the selection metric**: The main server sends a list of potential mesh servers to the client. The client measures the end-to-end delay to each mesh server using probing packets, then selects two of them with the minimum end-to-end delay as primary and secondary.

3. **Distributed scheme with hop count as the selection metric**: The client floods the WMN with a request message for content. Each mesh server that possesses it replies to the client with the minimum hop count to the client's router.

4. **Distributed scheme with a routing metric as the selection metric**: The WMN uses the ETT routing metric. The path with minimum path ETT cost is used by the routing protocol. The client floods a request message, and then each replying server obtains the path ETT cost to the client's associated router. The path with minimum ETT path cost is returned to the client's mesh router.

The observed shortcomings found in the UPAC system are: (*i*) Client peering can be feasible if the network's population is high. However, the authors consider peering with neighboring clients, which means a small number of clients and at the same time assumes that one or more clients will have the same requested content; (*ii*) The first and second methods for server discovery and selection depend on a centralized server, which is neither scalable nor robust against failure; and (*iii*) The third and fourth methods for server discovery and selection use flooding as a way for locating content. However, this exhausts the network's bandwidth injecting extra traffic.

The work in [8] proposed two architecture designs for MeshCache: (1) A2, where caching occurs at the client's access mesh router upon file download; and (2) A3, where caching occurs at each mesh router along the route in a per-hop transport fashion. They designed cache selection protocols for efficiently locating caches containing data objects for these two architectures.

1. **Cache selection protocols for architecture A2**: MeshCache selects a suitable MR to retrieve the needed object. The design choices used for that are:

   (a) *Tree-based Hierarchy Cache selection Protocol (THCP)*: This is the default scheme in which the access MR simply routes the request to the GMR (gateway) in case of a local miss. The access MR selects the best GMR by querying the routing protocol (probing of link metrics). Later on the object is transported and cached at the access MR and served to the requester.

   (b) *Broadcast Cache selection Protocol (BCP)*: THCP only exploits local hits at the access MR and the GMR, but not MRs in the vicinity. BCP searches MRs in the vicinity. On a local miss, the access MR queries the routing protocol for the path metric ETX to the closest gateway. Then it broadcasts a UDP message to locate the content (inserting the metric X, a search path metric Y = 0 and a unique sequence number). Each node rebroadcasting the message adds the ETX value to the node it received the packet from into Y. If the value of Y exceeds X, then no need to search further because the requester has a better path to the GMR. Nodes that have a local hit reply to the access MR, which in turn then queries its routing protocol to find a node with the best path metric.

2. **Cache selection protocols for architecture A3**: The difference between the architectures A2 and A3 is that when content flows back to the access MR in A3, it is cached at nodes along the way in a per-hop transport, which optimizes the cache selection protocols in A3.

   (a) *Per-Hop THCP (PH-THCP)*: In this scheme, on a local miss, selects the best gateway GMR and then finds the next hop node for GMR by querying the routing protocol. For example, in Fig. 3.4a (reprinted from the authors' paper), when MR4 has a miss it contacts MR2, which repeats the process and on a miss forwards the request to the next hop towards GMR.

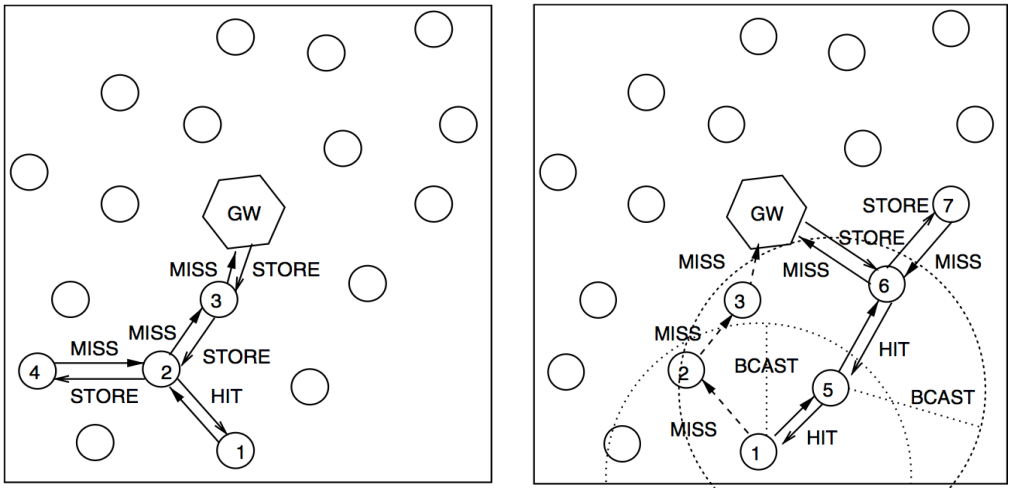
   (b) *Per-Hop BCP (PH-BCP)*: The only difference with BCP is that the content is transferred via per-hop transport from the selected MR. Also, if no hits occur or the metrics for all the hits are worse than the gateway, the access MR reverts to using PH-THCP to forward the request to the gateway. For example in Fig. 3.4b (reprinted from the authors' paper), MR7 has a local miss and no hits from the broadcast search. Then using PH-THCP, it fetches the content from the IGW via MR6.

MeshCache demonstrates throughput improvement from routing away from a gateway by alleviating the congestion around the gateway. However, the observed shortcomings that we found are as follows:

1. Caching the data at every mesh router along the path to the requester may result in the routers being flooded with redundant data.

2. PH-THCP does not consider content lookup from the neighboring nodes.

3. PH-BCP uses flooding to discover the cache location, which adds network overhead and bandwidth consumption.



(a) PH-THCP (b) PH-BCP

Figure 3.4: Cache selection protocols for architecture A3.

In [72], the CacheRescue takes advantage of the MRs storage capacity, since MRs are the Cluster Heads (CH) of the clusters formed by Mesh Clients (MC) associated with it. When a valid data item is replaced from MCs cache, it is sent to its CH for caching it in the CacheRescue Database (CRDB) until its TTL elapses, and then it can be evicted. When the CRDB becomes full, no more data items to be admitted, until there is enough free space (i.e., when some data items expire). The MeshSynch is a module that enables CHs to autonomously exchange and synchronize their cluster database with each other to make up a Network Database (NetDB), which increases availability, accessibility, and resource utilization. The MeshSynch continuously monitors the traffic state and invokes the MeshSynch process when the traffic is low. MRs exchange and synchronize their Cluster Cached List (CCL) from the Cluster Database (CDB) with each other to update the NetDB. If the CCL has updated entries, it sends the updates to other MRs as multicast messages. On receiving the CCL, each MR updates its NetDB. MRs maintain both the Cluster Cached List

and Network Cached List. CacheRescue increases availability, accessibility, and resource utilization. Moreover, each CH contains its copy of NetDB, which facilitates accessibility. However, it introduces a high cost in terms of space and computation overhead, since each CH has to maintain the NetDB. Furthermore, it adds a heavy traffic burden on the network caused by exchanging the CCLs of CHs.

The authors in [73] proposed SCAP (Smart Caching in Access Points) caching mechanism to reduce the upstream traffic in P2P-based live streaming. The mechanism aims to quickly identify the downstream packets that will be uploaded to or relayed to other peers. Instead of uploading the entire data packet, SCAP temporarily caches the corresponding downloaded packets in the AP, and the relay peer only uploads a small identity tag to the AP, reducing the upstream traffic in the WLAN. The work in [74] proposes a Network Assisted Peer-to-Peer (NAP2P) system for file-based media streaming services such as video-on-demand in WMNs in which mesh routers dynamically cache content and form a P2P network with end user devices. The system enables efficient and scalable communication mechanisms, such as automatic caching and multi-source multi-path streaming to optimize system performance and to meet the QoS requirements of streaming sessions.

3.5 Replication Schemes in the Internet

Related work on replicating content has mostly concentrated on the problem of placing the replica servers for one origin server. In our work, we consider a more global case, where we have content from several origin server and we decide which objects to replicate and place them on potential replica servers. Table 3.1 lists the major differences between replica server placement and replica content

placement. The replica placement problem can be expressed in two forms [75]:

Table 3.1: Differences between replica server/content placement.

Replica server placement	**Replica content placement**
Aims at selecting good locations for hosting replicas of many objects.	Aims at selecting locations that are good for replicas of a single object.
It happens in a larger time scale (i.e., once every few months)	Runs more often as it needs to react to rapidly changing situations such as flash crowds.

1. **Replica server placement**: Is the problem of finding suitable locations for replica servers.

2. **Replica content placement**: Is the problem of selecting replica servers that should host replicas of an object.

The work in [76] proposed a Web proxies' placement algorithm based on the assumption that the underlying network topologies are trees and solved it using dynamic programming techniques. It works by dividing a tree T into smaller subtrees T_i. The algorithm is shown to be optimal if the underlying network topology is a tree. However, this algorithm has the following limitations: (*i*) It cannot be applied to a WMN whose topology is not a tree (generalization problem); and (*ii*) It has a high computational complexity of $O(N^3K^2)$, where K is the number of proxies and N is the number of candidate locations.

In [77], three approaches have been proposed for the Web Server replica placement:

1. **The greedy algorithm**: They model the Web server replica placement problem as a minimum p-median problem and propose a greedy algorithm.

At every iteration, the algorithm selects one server that offers the least cost. In the i^{th} iteration, it evaluates the cost of hosting a replica at the remaining $N - i + 1$ potential sites in the presence of already selected $i - 1$ servers. The computational cost of the algorithm is $O(N^2K)$.

2. **The random algorithm**: This algorithm is oblivious to the client workload, where it picks randomly M replicas among N potential sites from a uniform distribution. The algorithm is executed several times and then chooses the set that yields the lowest cost.

3. **The hotspot algorithm**: This algorithm places replicas near the clients generating the greatest load. It sorts the N sites according to the amount of traffic generated within their vicinity. It places the replicas at the top K sites that generate the largest amount of traffic. The computational complexity of the hot-spot algorithm is $O(N^2 + min(NlogN, NK))$.

They found that the greedy algorithm performs very well in practice, typically within a factor of 1.1 to 1.5 of the optimal solution. However, our work considers the popularity of an object as an important factor that reduces the access cost. They assume that the replicas are complete replicas and they do not study replicating individual objects; in our work, we consider replication per-object granularity.

The work in [78] proposed the placement of intercepting proxies known as TERC; they present different placement strategies that can be used to reduce the network traffic or the average access latency. They propose optimal solutions for simple topologies, such as line and ring, and consider the case of placing proxies for a single server in a tree topology. However, they assume that caches are to be placed only en route between a client and the server, while our placement schemes assume that replicas can be placed on any node in the WMN.

The authors in [79] used graph theoretic approaches and heuristics for placing Internet instruments to obtain distance maps. They found that Internet distance map using their placement techniques can be used by clients for server selection. The objective function that the authors propose is to place the tracers optimally such that the maximum distance from any client to its nearest tracer is minimized. We note here that: (*i*) The heuristics are based on the p-center problem definition that considers the distance only, ignoring the clients' demands; and (*ii*) They consider that some fixed nodes are selected as replica servers, while others only act as clients, contrary to our system that considers a P2P schema, where a node may act as content replica server or a client, which improves load balancing in our system.

The work in [80] performed a broad evaluation of different replica placement algorithms. Their findings were that web caching is more efficient if the update is highly frequent. However, for our approach in content delivery, replica placement algorithms are very important to guarantee object availability, prefetching, reliability, insured consistency, and performance. In [81], the authors proposed two topology-informed placement algorithms for replica servers regardless of knowing clients' locations. Based on the network topology alone, the first algorithm selects the servers that are close to the router with the maximum fanout in the network, while the second algorithm selects the Autonomous System (AS) having the highest fanout, and then selects a server within the selected AS closest to the router having maximum fanout. This approach relies on the node's degree expecting that nodes with higher degrees can reach more nodes with less latency. However, these algorithms assume a uniform distribution for clients in the network, which may not be true. However, if the assumption does not hold, then it might select servers near highest fanout routers but far from most of the clients resulting poor client perceived latency.

An algorithm was proposed in [82] that arranges the replica servers storing replicas of the same object into a load-balanced tree. When the origin server receives clients' requests that contain latency constraints, it serves the requests if the server's capacity constraints and the client's latency constraints are met. If any of the conditions fail, it will search for another server in the tree that satisfies both constraints and creates a replica at that server. The algorithm performs well at preserving client latency and server capacity constraints. However, it has a considerable overhead caused by checking QoS requirements for every request. A single request might result in creating a new replica, which will increase the request service time.

In [83], the authors modeled the content placement as an optimization problem, which is to place J objects in I potential servers. The solution aims to minimize the average number of inter-AS hops a request must traverse to be served, while meeting the storage constraints of each server. They have shown that the problem is NP-complete and proposed four heuristics (*KRR*[1] heuristics) for object replica placement of different origin servers:

1. **Random**: This heuristic assigns object replicas to the storage nodes in a random way subjected to the storage constraints. It works by picking one object with uniform probability and one node with uniform probability then stores the selected object in the selected node. If the node has already stored the object, then it randomly picks a new object and a new node resulting that an object could be replicated in several nodes. However, a node will have at most one object replica.

2. **Popularity**: In this heuristic, each node stores the most popular objects among its clients by sorting the objects in decreasing order of popularity,

[1]For brevity, we refer to the four heuristics as *KRR* after the authors' names.

then stores as many objects as the storage constraint allows. The node estimates the objects' popularities by monitoring the requests it receives from its clients. This heuristic does not require the node to get any information from outside the node.

3. **Greedy-Single (G-S)**: The cost function is defined as the product of the object popularity and the distance from the replica server to the origin server. Each node i computes $C_{ij} = p_j d_{ij}(x_0)$, where $d_{ij}(x_0)$ is the distance from node i to the origin server hosting object j. Then the node sorts the objects in decreasing order of C_{ij} and stores as many objects in this order as the storage constraint allows. The popularity p_j is obtained as in the *Popularity* heuristic, it also requires information about the network topology to estimate different d_{ij}s. The costs (C_{ij}s) are calculated only once under the placement x_0 at the origin servers and not adjusted when objects are stored. However, every node stores objects independently of other nodes and no cooperation between nodes is needed.

4. **Greedy-Global (G-G)**: It employs a globally coordinated replication strategy for all objects. The cost function $C_{ij} = \lambda_i p_j d_{ij}(x_0)$, which is the product of total request rate for a server node (λ_i), popularity and shortest distance of a server to a copy of object j. Then the central server picks the node-object pair that yields the highest C_{ij} and stores the object in that node resulting in a new placement x_1. Then the central server re-computes the costs C_{ij} under the new placement and picks the node-object pair that yields the highest cost and stores the object in that node obtaining a new placement x_2. The process keeps iterating until all the nodes' storage have been filled.

Upon evaluation in their own simulation, the authors found that *Greedy-Global* outperforms the other three heuristics. However, their model is oblivious about the object size, while we assume variable object sizes. Moreover, it is centralized and incurs high computation and communication costs.

In [84], the authors proposed *Lat-CDN*. The main idea is to assign the objects to replica servers with respect to the total network's latency that is produced according to objects' placement, without taking into account the objects' popularity; hence, the distance reflects the latency. Initially, all the objects are stored in the origin server(s) and all the CDN's replica servers are empty. For each outsourced object, it finds the best replica server in order to place it (i.e., produces the minimum network latency). Then, it selects from all the pairs of $\langle$outsourced object replica server$\rangle$ that occurred in the previous step and the one that produces the largest network latency, and thus assign this object to that replica server. This process is iterated until all the replica servers' storage become full. As a result, an object might be assigned to several replica servers. However, this algorithm does not consider the object's popularity, instead, it considers the placement on the replica server that gives the highest latency. What if the object was not popular in the selected server? This might lead to improper allocation of $\langle$replica object, replica server$\rangle$. Moreover, it is centralized and has a high complexity of $O(N^2K)$.

The authors in [85] modeled the object replica placement problem in P2P networks as a Clustered p-center problem. They propose an approximation algorithm to the problem with a provable upper bound. For load balance purpose, nodes are required to keep roughly the same number of objects across the network, which may help to ensure the queries are evenly distributed among the nodes. However, the p-center problem is different than the p-median problem since the p-center problem considers minimizing the maximum distance to access

an object, which can be very beneficial in terms of load balancing. On the other side it might lead to a placement that does not consider the nodal demand for that object.

In [86], the CNF architecture is based on a network infrastructure for hop-by-hop store-and-forward of large objects. It works by storing the object at the CNF and forwarding it to the next-hop CNF. When received, it is stored at the CNF router that acknowledges the previous CNF router, which deletes the stored copy from its store. The basic component is the CNF router that has persistent storage to hold objects in transit for potentially long periods of time. The main concepts of the architecture are listed below:

1. **Post Office (PO)**: A sender places the object to be delivered in its PO and the network routes it to the receiver's PO that holds the object until it is delivered to the receiver.

2. **Cache and Forward (CNF) router**: The CNF router works in a hop-by-hop store and forward fashion from the sender's PO towards the receiver's PO using forwarding tables updated by a routing protocol.

3. **Cache and Carry (CNC) router**: The CNC Router is a mobile network element that has persistent storage exactly as in a CNF Router, but is additionally mobile. It can pick a package from a CNF, CNC or from a PO and carry it along. It may deliver the package to its receiver or another CNC.

However, the architecture does not cache or replicate the object; instead, it only provides a reliable and acknowledged content delivery to the consumer. The focus in [87] is on server replication schemes for distributed systems that satisfies both of *push-based* and *interactive* properties. They propose a protocol for replica

server selection such that the network and user satisfaction costs associated with objects' updates are as low as possible.

A hierarchical content routing architecture for large-scale CDN was proposed in [88], where CDN servers perform intra-cluster and inter-cluster content routing based on a two-level hierarchical overlay network. Using qualitative analysis and simulations, they show that the scheme incurs small routing overhead and high content sharing efficiency. In [89], the *Community Influenced Caching* (CIC) algorithm was proposed for large-scale structured P2P systems. It aims to seamlessly and proactively cache resources for individual communities that are of interest to them. Therefore, a popular content lookup can be resolved faster within a community that shares the same interests. These communities form sub-overlays within the overlay network. The replication decision is based on the community's interests. However, a community's sub-overlay peers need not to be in each other's proximity. This assumption is suitable for replication in the wired Internet where there is ample bandwidth. Our approach is different since we aim to minimize the demand-weighted distance to fetch an object. In [90], the authors propose a replication strategy to find the optimal replica distribution to obtain the minimum expected search size. The strategy is composed of two parts: The first part gives the optimal replica distribution under heterogeneous environment, where the object size is considered to be an important factor; the second part comes up with an optimal distribution under the homogeneous environment.

Table 3.2 summarizes and classifies the different replication and caching schemes showing the pros & cons of each, and the differences with our proposed replication schemes.

Table 3.2: Classification and comparison of different replication schemes.

Ref	Main Feature	Pros	Cons and differences from our work
Wireless Mesh Networks			
[47]	p-center problem, maximize distance between identical replicas	Fully distributed, scalable & asynchronous.	Content popularity is not considered.
[38]	P2P file sharing	Selects multiple providers for a requested file.	Does not address object replica placement.
[69]	Popularity based placement	Determine the optimal number of replicas.	No cooperation between nodes might yield to the placement of identical replicas in the vicinity.
[37]	Improve P2P DHT	Speedup content lookup	Does not research content placement
[39]	Improve P2P FTP	Select download peers with least-loaded paths	Does not research content placement
[40]	Place highest popular clips on "all" MRs	Maximizes No. of peers	Does not consider node's popularity
[41]	Minimize system failure rate	Improves robustness against failure	Does not consider clients demands
[42]	Placement at a predefined set of nodes	Minimize access cost	Assume equal service demand
[68]	Opportunistic caching	Caches passing by and/or overheard chunks	Cache redundancies
[71, 74]	Unified P2P & cache	Reduce server load	Does not propose a placement approach

Continued on next page

Table 3.2 – continued from previous page

Ref	Main Feature	Pros	Cons and differences from our work
[8]	Caching architecture	Improve access latency	En-route cache redundancies
[72, 73]	Cache clusters	Increases availability, accessibility & resource utilization	High cost (space & computation overhead)
Wireline Internet			
[83]	Greedy placement based on popularity & distance	Minimize object access cost.	Centralized, incurs high computation & communication cost.
[87]	Full server replication	On-the-fly server replication when it is overwhelmed with client's requests	Replication is driven by server capacity status rather than popularity change.
[76]	Web proxies' placement algorithm	Optimal for tree topology	Limited by tree topology
[77]	Server replica placement	1.1 to 1.5 of optimal solution	Replicating individual objects
[78]	Intercepting proxies	Optimal solutions for simple topologies	Does not apply to WMNs
[81]	Topology-informed server placement	Servers are close to routers with maximum fanout	Assume uniform clients distribution
[85]	p-center problem	Clustering approach	Content popularity is not considered.
[82]	Load balancing	Improves load distribution	Does not propose a placement approach
[84]	Object replica placement	Minimize the maximum distance	Does not consider content popularity

Continued on next page

Table 3.2 – continued from previous page

Ref	Main Feature	Pros	Cons and differences from our work
[86]	Hop-by-hop store-and-forward	Reliable & acknowledged content delivery	Does not cache/replicate the object
[87, 89, 88, 90]	Replica server selection	Reduce object update cost	Does not propose content placement
Mobile Ad hoc Networks			
[46]	Disseminate object replicas on nodes at r-hop distance	r-hop depends on replication degree	Does not consider popularity in placement decision
[48]	Content replicas roam the network	Simple approach	Replica node selected randomly
[49]	Load balancing according to variations in network topology & query rate	Distributes burden of storing & providing content on nodes to achieve load balancing	Do not replicate the content according to clients' demands
[50]	Conditional betweenness centrality	Use of social networking concepts	Content is not replicated over multiple nodes
[56, 59]	Zone-based search	Improve cache lookup	No cooperation between caches beyond each one's zone
[57]	Neighbor Caching	Utilize cache space of inactive neighbors	Does not utilize an efficient cooperative caching
[60]	GroupCaching	Caching locally or randomly at a group member	Cache redundancies
[61]	Cache resolution schemes	Improve data availability	Incurs heavy traffic overhead

Continued on next page

Table 3.2 – continued from previous page

Ref	Main Feature	Pros	Cons and differences from our work
[62]	Cooperative caching system	Efficiently find cached objects	Does not research content placement
[64]	Cooperative caching techniques	Reduce access latency	Not fully cooperative
[65]	En-route caching	Minimize object copying overhead	Content lookup considers only en-route nodes
[53, 54, 55]	Replication in dense networks	Maintain a fixed replication degree	Does not propose content placement
[44]	Incentive based scheme	Gives high-quality multimedia to premium users to provide it to ordinary users	Content selection is dominated by premium users
[58, 66, 67]	CCN-based	In-network caching, discovery & delivery	Uses the simple LFU caching policy. Different from replication.
[51]	Content sharing based on BitTorrent	Content discovery reduces routing overhead	Does not address object replica placement
[52]	Data lookup & replication	Replicate far & highly requested items	No cooperation, yields to placement of identical replicas in the vicinity
[63]	Network deduplication	Reduce redundant traffic packets that improves network performance	Does not address object replica placement

3.6 Summary

We can summarize the literature review as follows: On one side, in WMNs and MANETs, the related work focuses on topics that improve content lookup services, cooperative FTP, unreliable and unplanned placement of content at home devices. Other approaches of content networking use different caching schemes such as Neighbor Caching, Group Caching, CacheData, CachePath, HybridCache, roadside caching, and conventional caching policies (i.e., LRU, LFU). Moreover, cache resolution and discovery are well researched such as flooding, profile-based, roadside discovery. On the other side, in the wired Internet, the related work can be classified into replica server placement and object replica placement. Our focus in this thesis is on the latter. Many algorithms have been proposed for the Internet [83], [84] and [85]. These algorithms consider the hop-count as the distance in the cost function to be minimized. However, in WMNs there are other routing metrics to be considered in the cost function, which can significantly improve the network performance. Another shortcoming to be considered is that these algorithms are centralized and incur a high computational/communication complexity, which might not be an issue in CDNs that have high performance servers. Implementing these algorithms in WMNs will require long running time due to the scarce resources and the dynamic wireless environment. To the best of our knowledge, [83] and [84] are the most relevant to our work as they both solve the placement problem in a P2P fashion and formulate the placement problem as an integer program to minimize the average travel time of the objects. Therefore, throughout this thesis, we compare our schemes with them.

Chapter 4

A Hierarchical Approach for Object Replication

One of the key challenges for the development of WMNs is improving the data access efficiency and Quality of Experience (QoE), which determines the satisfaction degree of service users. Furthermore, as the Internet traffic flows through a limited number of IGWs, heavy congestion around these IGWs presents a serious problem. In WMNs, the contention between neighboring mesh nodes for the wireless channel, together with the interference from the adjacent wireless links, results in a significant reduction in throughput over a long path. Therefore, MRs that are far from the IGWs suffer from long access latency and low throughput.

WMNs have the potential to increase network capacity by adding resource-sharing services such as content caching and replication. It has been observed that, for a given client population, a significant workload *locality* exists in Web traffic over both Internet [6] and WMNs [8] content retrieval. Locality means that multiple users request the same content over time (and possibly at the same time). Web content caching and replication are two techniques that exploit locality to reduce the Internet traffic and access latency.

Our contribution in this chapter proposes the *SP-DNA* (Single Partition per Delegate Node Assignment) replication scheme for WMNs that builds an overlay P2P network formed by MRs. The scheme dynamically adapts the number of object replicas over time based on popularity. *SP-DNA* is a distributed and scalable scheme that exploits workload locality in WMNs by finding the number of replicas needed per object (e.g., video file for a Video-on-Demand service) in a time window, and takes into account the variation of content popularity over time. The scheme uses graph partitioning techniques to distribute the replica placement problem on the *Delegate Nodes* (DNs). A DN is responsible for replica placement within its partition. The challenge of minimizing object access cost is more serious in WMNs than the Internet. Minimizing the hop-count may not be an issue in the wired network, as long as there is sufficient bandwidth between the requesting client and the replica server. In WMNs, the contention between neighboring mesh nodes for the wireless channel, together with the interference from the adjacent wireless links, results in a significant reduction in throughput over a long path. Therefore, MRs that are far from the IGWs suffer from long access latency and low throughput. We consider minimizing the demand-weighted distance between the requesting node and the replica server.

4.1 Our Proposed SP-DNA Scheme

In this section, we present a proposal that is focused on decomposing the replica placement problem by using graph partitioning techniques. The main goal of graph partitioning is to divide a graph into a set of sub-graphs such that each sub-graph has roughly the same number of nodes and the sum of all edges that connect different sub-graphs is minimized. Therefore, graph partitioning is useful to distribute the problem into a set of partially independent sub-problems

Table 4.1: Notation used in the proposed schemes throughout the thesis.

Symbol	Meaning
$\mathcal{N}'$	Set of demand nodes within a partition indexed by i', and the set of potential replica servers indexed by j'.
$\mathcal{K}$	Total number of the generated graph partitions indexed by k.
$\mathcal{DN}_k$	Delegate node of partition k.
$\overline{X}_t$	Trimmed mean of the link-quality metric (e.g., ETX, ETT and ML).
$\Gamma(\overline{X}_t)_{i'}^{j'}$	Distance between demand node i' and potential replica server j' within $\mathcal{N}'$ as a function of $\overline{X}_t$.
τ	Time interval window during which, object requests are observed.
$\lambda_{mi}(\tau)$	Number of requests for $\mathcal{O}_m$ from node i during τ.
$\lambda_m(\tau)$	Total number of requests for $\mathcal{O}_m$ from all nodes during τ.
$\mathcal{P}r_m(\tau)$	Global popularity of $\mathcal{O}_m$ during τ.
$\mathcal{P}r_{mk}(\tau)$	Popularity of $\mathcal{O}_m$ during τ within partition k.
$\mathcal{P}r_{mi}(\tau)$	Popularity of $\mathcal{O}_m$ at node i during τ.
$p_m(\tau+1)$	No. of replicas of $\mathcal{O}_m$ needed for $\tau+1$.
$d_{i'j'}$	Distance between demand node i' and potential serving node j' within partition $\mathcal{N}'$.
$\mathcal{SC}$	Storage capacity of a node.
ζ_{mk}	Current no. of replicas for $\mathcal{O}_m$ within partition k

especially when dealing with large problems. The search space is split according to the computed number of replicas per object. In each partition, a predetermined *Delegate Node* (DN) solves part of the complete search space. To this end, the replica placement problem is divided into a set of smaller loosely connected ones. Thus, we can take advantage of graph partitioning techniques to solve the problem considered in a divide-and-conquer approach.

To distribute the replica placement problem, we simplify the p-median problem by partitioning the network graph into p sub-graphs, where p represents a potential number of replicas. Then we select for each partition a DN, which will be responsible for placing an object replica within its partition. The partitioning algorithm we use is proposed in [91] and has been implemented by Metis [92] software. In our scheme, we recursively bipartite the graph until the partition size ≤ 2. Since the total number of partitions = $\mathcal{N} - 1$, each DN will be assigned at most a single partition; hence, we call it *SP-DNA*. Our scheme involves two phases, the *Network Setup Phase* and the *Content Replication and Placement Phase*. We use the notation in Table. 4.1 to describe our schemes throughout this thesis. Therefore, some symbols in the table are used in the next chapters but are not used in this chapter.

4.1.1 Network Setup Phase

In this phase, the aim is to build a hierarchy of partitions and assign a DN for each partition to lookup for the placement within the partition's member nodes. We assume that all nodes[1] are bootstrapped with Alg. 1. To trigger the network setup phase, an application process called the Content Manager (CM) *Content Manager* (CM)—which is a component of the *ReplicaDaemon*—starts this phase.

[1]In order to avoid confusion throughout this thesis, the terms MRs, nodes and $\mathcal{N}$ refer to the same meaning since our system is implemented by MRs decoupling it from MCs.

To reduce the message overhead, the CM can be hosted by a centroid node. The CM plays a role in collecting statistical information about different objects' requests, computing the number of replicas per object for the next τ period, and assigning the placement job to the corresponding DNs.

We define two graph structures g_{Left} and g_{Right} in line 3 since each DN will bipartite the received graph G into two partitions (sub-graphs) and selects one of the member nodes to act as a DN for each partition; hence, we define two node structures dn_{Left} and dn_{Right} in line 3. The two string variables s_{Left} and s_{Right} (line 4) are used to keep track of the already selected DNs. Alg. 1 is triggered by the CM with G representing the whole graph and the string variable s initialized with the NULL string. The CM selects itself to act as a DN for the network graph G and initializes both s_{Left} and s_{Right} to the value of s (line 6, which is a NULL string when the instance is called by the CM. The CM bipartites G (line 20, selects a DN for each partition (lines 20 and 21) such that the selected DN was not assigned before to another partition. This is verified by making sure that the selected $dn_{\text{Left}} \notin s_{\text{Left}}$ and $dn_{\text{Right}} \notin dn_{\text{Right}}$. Afterwards, we append (concatenate) dn_{Left} and dn_{Right} to s_{Left} and s_{Right} respectively (lines 22 and 23).

The CM then forwards each partition (sub-graph) to its corresponding DN (lines 25 and 26) and waits (line 27) for its child DNs to return the string representation of the progenies of each branch. This process is recursively repeated until the partitions are fine-grained. When the base case is reached (line 6), the partition size is 2 the node running the algorithm instance selects itself as the left DN (the term *this* refers to the node itself (line 7)) and selects the remaining node (line 8) to be the right DN appearing as leaf nodes in the generated tree. Note that when the network size is a power of two, the case in line 11 will not be satisfied. Otherwise, at some stage, the partition size will eventually be 3. In this case, it selects one of the nodes to be dn_{Left} (line 12) to act as a leaf node. The

Algorithm 1: This function builds the DBT during the network setup phase.

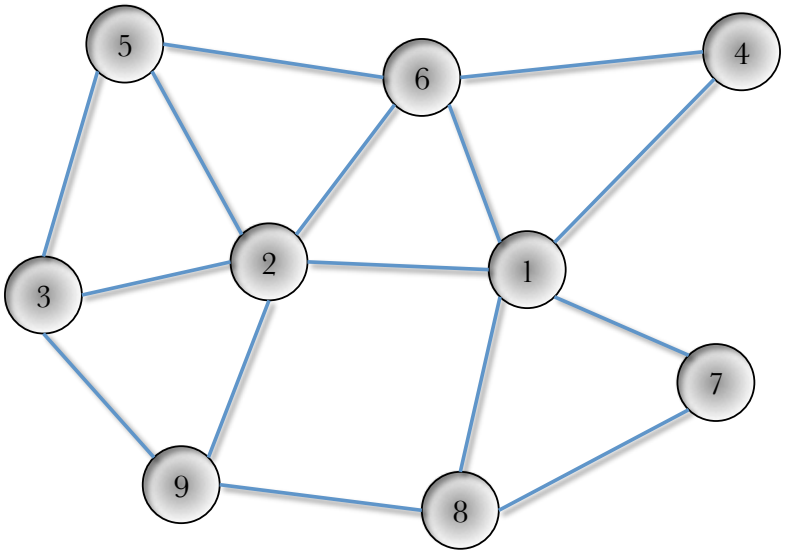
Input: Network graph G.

Output: A string representation of the DBT returned to the CM.

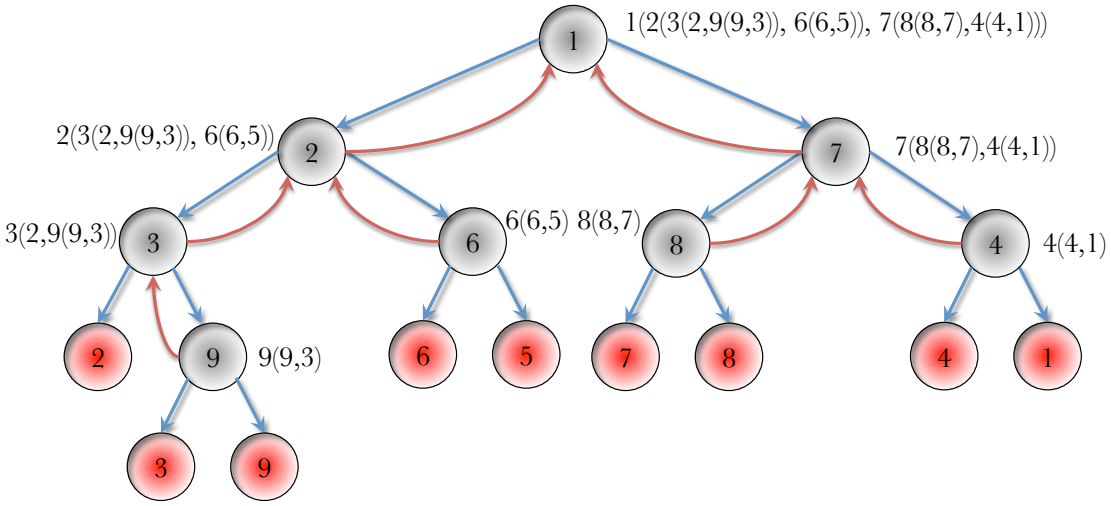
```
1  Function BTree(Graph g, String s)
2  Graph g_Left, g_Right
3  Node dn_Left, dn_Right
4  String s_Left, s_Right
5  s_Left ← s_Right ← s
6  if |g| = 2 then
7      dn_Left ← this
8      dn_Right ← {g} \ dn_Left
9      return dn_Left || {g}
10 end
11 else if |g| = 3 then
12     dn_Left ← rand(g) ∉ s
13     g_Right ← {g}\ dn_Left
14     dn_Right ← rand(g_Right) ∉ s
15     forward (g_Right, s) → dn_Right
16     wait (dn_Right)
17     return this || dn_Left || dn_Right
18 end
19 else
20     Bipartite (g, g_Left, g_Right)
21     dn_Left ← rand(g_Left) ∉ s_Left
22     dn_Right ← rand(g_Right) ∉ s_Right
23     s_Left ← s_Left || dn_Left
24     s_Right ← s_Right || dn_Right
25     forward (g_Left, s_Left) → dn_Left
26     forward (g_Right, s_Right) → dn_Right
27     wait(dn_Left, dn_Right)
28     return this || dn_Left || dn_Right
29 end
```

remaining two nodes form the right partition (line 13) and then selects a dn_{Right} (line 13) to act as a DN for a partition of size 2.

The DNs in the lowest level partitions reply (line 9) to their callers (parent node) with a list representing the DN and its partition members. When the caller receives the lists from its child DNs, it merges the received lists (lines 17 and 28), appends its node ID and forwards the resulting list to its caller. Eventually, the CM (root node) will receive a list of DNs in the form of a balanced binary tree that we call the *Delegates Binary Tree* (DBT) as depicted in Fig. 4.1. Notice (Fig. 4.1b) that a node appears twice. Once as a parent node (i.e., partition-hosting) responsible for the descendant members (and itself) and another as a leaf node (i.e., self-hosting). We underline that this phase needs to be performed once. However, if the topology changes permanently (e.g., relocation of MRs), then we need to perform this phase again. We do not consider a temporary topology change due to temporary link variations, as content replication is required for long periods. Fig. 4.2 illustrates the generation of partitions for potential number of replicas and the assignment of a DN for each partition.



(a) Topology view.



(b) Tree view.

Figure 4.1: This figure illustrates the conversion of the network graph G (a) into a balanced binary tree DBT (b), where each node represents a DN for the underlying partition.

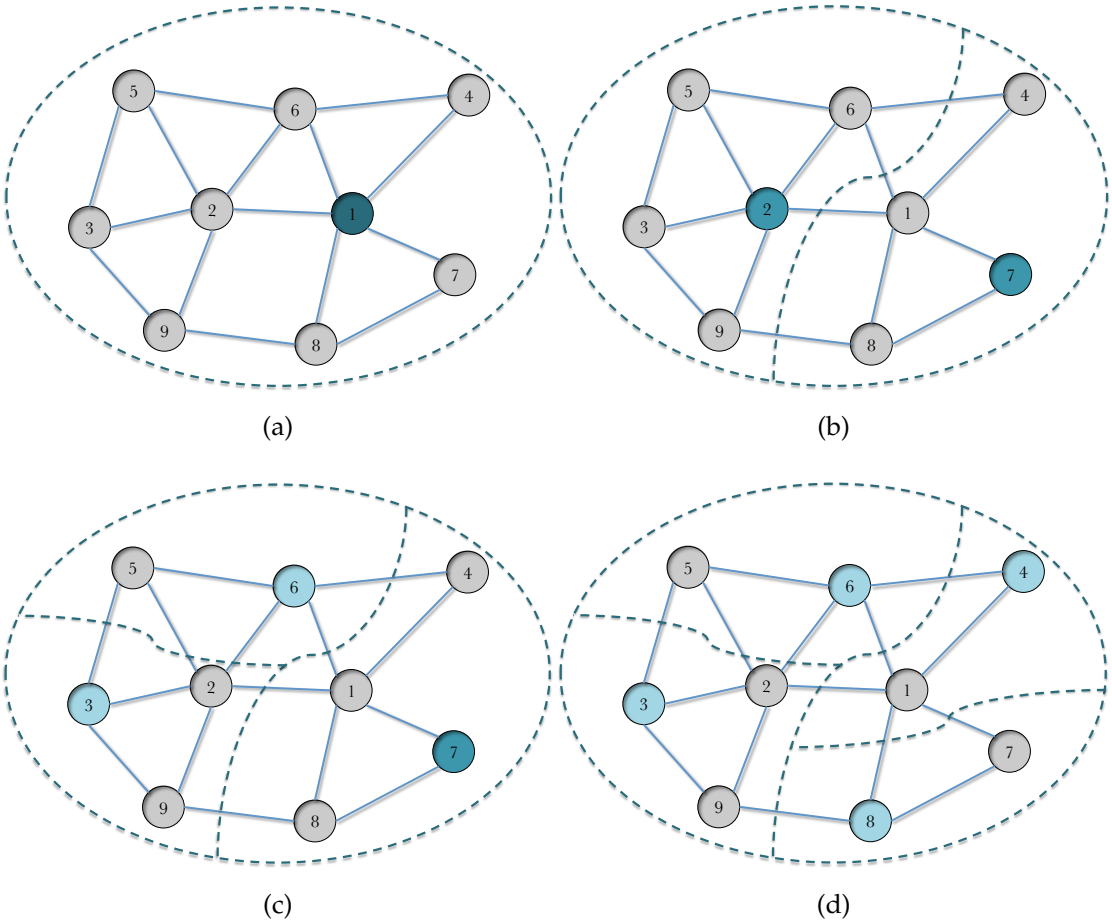


Figure 4.2: This figure depicts the first 4 stages of the Network Setup Phase, where each stage represents the generation of (semi) equal partition sizes and assigning a DN for each partition.

Following the DBT creation, the CM maps the possible numbers of replicas to their DNs in the DBT, creating a Map-List by running Alg. 2. It recursively divides the possible number of replicas (p) by 2 starting from the root node. When the base case is reached (i.e., $p = 1$), it checks whether the node n is a leaf node in the DBT or not. If it is a leaf node, then it means that node n has to *self-host* the object for p replicas and returns the node ID concatenated with a flag (S_{host}) to indicate *self-hosting*. Otherwise, it returns the node ID along with a flag (P_{host}) to indicate *partition-hosting* (i.e., the DN will need to find the optimal location for the object within its partition). For every possible p, the CM creates an entry in the form of $\langle p$, set of DNs$\rangle$ and inserts it in the Map-List structure. Upon completion, the CM broadcasts the Map-List to all the nodes. This will facilitate for both the CM and the DNs to know textita priori that for a specific p, which DNs the CM shall communicate with to assign the placement job and the DN knows from the Map-List its role whether self-hosting or partition-hosting.

Algorithm 2: Function to map the possible number of replicas (p) to their DNs.

```
1  Function Map(Node n, int p)
2  if p = 1 then
3      if n is leaf then
4          return n||S_host
5      else
6          return n||P_host
7      end
8  else
9      return map(n.left, ⌊p/2⌋) + map(n.right, ⌈p/2⌉)
10 end
```

4.1.2 Content Replication and Placement Phase

This phase consists of three steps to be performed aiming to collect statistical information about the objects, compute the number of replicas for each object and then find the placement for the replicas. These steps are described as follows:

During the τ period

During the replication period τ, every node maintains a list containing information about the request frequency observed by the node. The list entries are in the form of a 4-tuple $\langle MR_i, \mathcal{O}_m, |\mathcal{O}_m|, \lambda_{mi}(\tau)\rangle$ that represents the request count λ_{mi} for every object $\mathcal{O}_m$ during τ at node i. An object request is counted when a MC initiates one to its access MR regardless of being served by the access MR, any other MR or even the origin server.

At the end of the τ period

In this step, the DNs send the collected statistical information to the CM that computes $p_m(\tau + 1)$. The sub-steps are described as follows:

a. The partition members of the lowest level DNs forward their object frequency list obtained from previous step to their DN. Each DN aggregates the received lists from its children along with its own list, stores the resulting list and then forwards it to its parent node. The process of aggregate, store and forward is repeated until the root node receives the full list for the whole WMN and then forwards it to the CM. The usefulness of this hierarchical approach is: (*i*) Reducing communication overhead; and (*ii*) Fusing popularity information at different levels of the DBT helps distributing this information instead of collecting it by a central node.

b. For every object $\mathcal{O}_m$, the CM computes the global popularity $\mathcal{P}r_m(\tau)$ according to Eq. (4.1).

$$\mathcal{P}r_m(\tau) = \frac{\sum_{i=1}^{\mathcal{N}} \lambda_{mi}(\tau)}{\sum_{i=1}^{\mathcal{N}} \sum_{m=1}^{\mathcal{M}} \lambda_{mi}(\tau)} \tag{4.1}$$

c. The CM computes the number of replicas $p_m(\tau + 1)$ for every object $\mathcal{O}_m$ using Eq. (4.2).

$$p_m(\tau + 1) = \frac{\mathcal{SC} \times \mathcal{N} \times \mathcal{P}r_m(\tau)}{|\mathcal{O}_m|} \tag{4.2}$$

d. The CM creates a *Replica-List* ($\mathcal{RL}$) that contains information about the objects' replicas. The generated list structure is in the form of a 3-tuple $\langle p_m(\tau+1), \mathcal{O}_m, |\mathcal{O}_m| \rangle$ grouped by $p_m(\tau+1)$ in a decreasing order and within each group, the objects are sorted by their size $|\mathcal{O}_m|$ in a decreasing order. The reason behind this way of sorting is that we give priority for the highly popular objects and then the priority is given to the larger objects. Within a group of objects that have the same number of replicas, objects will have semi-equal popularity; prioritizing large objects would minimize the cost weighted by the object size given the storage constraint. However, our scheme is fair with small objects in terms of the computed number of replicas, since it divides the popularity of an object by its size (see Eq. (4.2)). This adopted approach of sorting was not considered in the schemes we compare our work with in this thesis and it cannot be adopted by such schemes. Therefore, they assume equal object size. In fact, our approach yields to different set of replicas and different number of replicas per object.

Replica assignment and placement

In this subsection, we describe our distributed replica placement heuristic (see Alg. 3). The heuristic has two sides. The first side is run by the CM and rep-

Algorithm 3: Pseudo code for the *SP-DNA* Heuristic.

foreach *group* $\in \mathcal{RL}$ **do**
    multicast a message containing the group of object replicas to the corresponding list of DNs in Map-List
end
upon receiving the message, a DN will do:
lookup the Map-List for the given $p_m(\tau + 1)$
if *the node is flagged s* **then**
    forall the $\mathcal{O}_m$ *in the received group* **do**
        `Fetch(`$\mathcal{O}_m$`)`
    end
else
    forall the $\mathcal{O}_m$ *in the received group* **do**
        select the node that minimizes the cost:
        $Min \sum_{i'=1}^{\mathcal{N}'} \sum_{j'=1}^{\mathcal{N}'} Pr_{mi'}(\tau) d_{i'j'}\ s.t \exists\ SC$
        assign $\mathcal{O}_m$ to the selected node
        `Fetch(`$\mathcal{O}_m$`)`
    end
end

resented by the first loop (lines 1 to 3). The second side is run by the DNs (lines 4 to 17). Based on the produced Replica List ($\mathcal{RL}$), the CM multicasts a message for each group in the $\mathcal{RL}$ to the corresponding DNs obtained from the Map-List in the *Network Setup Phase*. The message multicast is performed using the Application Level Multicast (ALM). This is carried out in a decreasing order of the number of replicas $p_m(\tau + 1)$ to prioritize objects with high popularity. The multicast message is in the form of a 3-tuple $\langle p_m(\tau + 1), \mathcal{O}_m, |\mathcal{O}_m| \rangle$ for each object in the group. Afterwards, each DN finds out from the Map-List whether it has to *self-host* or *partition-host* the objects in the received multicast message. If the node has to *self- host* (line 6) the listed objects, then it only needs to call the fetch function (line 8) to bring the needed object and store it. The description of the fetch function (Alg. 4) is discussed below. Otherwise (line 10), the case will be partition-host. The DN computes the total cost (line 13) for every partition member and assigns the object $\mathcal{O}_m$ to the node that minimizes the total demand-weighted cost. The distance $d_{i'j'}$ represents the shortest distance between i' and j'. Note that the selected node must have enough storage for $\mathcal{O}_m$. Otherwise, the DN picks the second best node and so forth constrained by the storage limit.

Upon successful assignment, the selected node fetches the object replica using Alg. 4 by trying to find it locally (line 2). Upon a miss, it consults the Directory Service (e.g., Chord [17]) by sending the object's key to obtain the list of nodes hosting it. If the object is found, the replica server requests it from the closest node. Upon a miss, it requests $\mathcal{O}_m$ from the origin server. We note here that lines 13 and 14 refer to two separate updates. The first (line 13) is for the replica server to differentiate objects of the next period $(\tau + 1)$ from objects of the current period (τ). The second (line 14) is used to update the Directory Service with the node ID hosting $\mathcal{O}_m$ for the next period $(\tau + 1)$. After fetching $\mathcal{O}_m$, the node evicts object(s) from previous τ and inserts $\mathcal{O}_m$ for $(\tau + 1)$. We can observe that the

Algorithm 4: Function to fetch an object performed by the replica server node.

```
1  Function Fetch(O_m)
2  if O_m is in the node's storage then
3  |   goto line 13
4  end
5  else if lookup(O_m) → the Directory Service then
6  |   receive the list of nodes hosting O_m
7  |   select the nearest hosting node for O_m
8  |   fetch O_m from the selected replica node
9  end
10 else
11 |   fetch O_m from the origin server
12 end
13 update O_m(τ + 1)
14 update the Directory Service with O_m(τ + 1)
```

SP-DNA heuristic is not fully distributed as it depends slightly on the CM that acts as a Super-Peer [93]. However, the major burden is on the replica placement, which is performed by the DNs. This avoids the fully distributed approach that incurs excessive message overhead resulting from the exchange of popularity statistics between all the participating nodes.

4.2 Evaluation Methodology

In this section, we describe our methodology for the conducted simulation experiments. We used OMNeT++ simulator [94] with Inetmanet [95] and OverSim [96] implementations. Inetmanet provides an implementation for different MANET routing protocols with the support for multiple link-quality metrics including hop-count, ETX, ETT and ML. OverSim is a flexible overlay network simulation framework based on OMNeT++. It includes several structured and unstructured P2P protocols. In our implementation, we use Chord [17], which is a well known distributed, scalable and DHT based content lookup protocol that is designed for structured P2P networks. We have implemented the *KRR* [83] heuristics[2] to compare them with our scheme since the problem considered is similar to the one in this thesis. Initially, we should create for each node its own subset of popular objects. Therefore, each node is assigned randomly 90% of the top 10% in the ranked pool $\mathcal{M}$ and the remaining 10% is selected randomly from the remaining 90% of the pool of objects $\mathcal{M}$. We should note that:

a. This assignment of *Objects of Interest* does not mean that these objects are *stored* in the nodes storage. However, creating this subset is important since each node will generate mesh clients' requests with a probability based on the ranked subset.

[2]We refer to the four heuristics as *KRR* after the authors' names

b. The generated requests are not limited to the *Objects of Interest* subset. This means that a high percentage (e.g., 80% or %90) of the requests are for objects that belong to this subset and a low percentage for objects in the rest of $\mathcal{M}$.

c. The contents of the *Objects of Interest* subset change at the end of every replication period τ based on the newly requested objects and/or objects belonging to this subset but received low demand.

Table 4.2: Default simulation parameters

Parameter	Default Value
Simulation area	800m x 800m
Number of MRs ($\mathcal{N}$)	100
Radio interface	IEEE 802.11g
Link rate	18 Mbps
Transmission range	140m
Zipf-like parameter α	0.95
Carrier frequency	2.4 GHz
Max. Tx power	20 dBm
Noise level	-110 dBm
Channel model	Rayleigh
$\mathcal{M}$	1000
$\|\mathcal{O}_m\|$ range	$1 \Rightarrow 4$ MB
τ	60 minutes
$\mathcal{SC}$	512 MB

Table 4.2 summarizes the default simulation parameters. We run the baseline *Random* heuristic for multiple of times and then use the same data set and generated requests to test the other *KRR* heuristics or the *SP-DNA* heuristic.

The simulation results were averaged over 10 multiple random scenarios. The simulation tests are carried out in a transient state system and online fashion such that object replica placement is performed whilst clients' requests are generated. For all the heuristics, requests are served from the closest replica server. The following models and simulation parameters' values are set during all simulation runs throughout this thesis unless we state different values in each simulation experiment.

a. *Network model*: We simulate a static WMN consisting of different numbers of MRs for different mesh topologies to investigate the efficiency and scalability of our placement heuristic. The network size considered is 100 MRs. The mesh routers were placed randomly in an area of 800m $\times$ 800m. The average distance between a MR and its neighbor is 100 meters. Each node is assumed to have one interface equipped with an omnidirectional antenna. All the nodes communicate with the gateway node to simulate the Internet access pattern.

b. *Client behavior model*: Each mesh router aggregates requests received from six active mesh clients on average. For each individual mesh client, a successive request arrives after the current request has been served. A mesh router may serve requests from different clients concurrently. The object request pattern of mesh clients is based on the Zipf-like distribution that models web traces as been found in [7]. The objects are assigned popularity probabilities based on the Zipf probability distribution. Values between 0.6 and 0.8 have been typically observed in Web proxy traffic [7]. Much higher values, up to 1.4, have also been discovered in the context of popular Web servers [97]. The value of the Zipfian parameter α was chosen to be 0.95 similar to [98]. The browsing behavior of the clients is simulated using a random think time period that represents the time elapsed between

successive web page downloads by a MC. This period ranges between 5 and 10 minutes on average since the requested object represents a multimedia Web content and not the ordinary Web page browsing. This is similar to the object request model in [64].

c. *Content model*: The set of distinct objects $\mathcal{M}$ was ranging from 500 to 3000 with an object size ranging between 1 and 4 MB following a Gaussian distribution from which each client makes a request based on the Zipf-like distribution. The application traffic is created using FTP traffic generator. We assume that content is in *Read-only* mode where write requests are not considered. Moreover, content consistency and lifetime is out of the scope in this thesis. The mesh router storage size $\mathcal{SC}$ is set to 512 MB. For content discovery, Chord protocol is employed, which is based on the idea of mapping both peer (mesh router) IDs and resource IDs (keys) into the same ID space. Requests are evenly distributed between MCs. To serve a request, a MC client sends the request to the MR it is associated with. The MR looks up the requested object locally, if found, it will serve it; otherwise, it will query the object's key to find out the node responsible for it. The responsible node in the chord ring overlay replies with a list containing the IP addresses of the hosting replica nodes. The requesting MR node consults the routing protocol to find the closest replica server using Dijkstra's algorithm that finds the shortest path to the serving node.

d. *Routing, transport and MAC*: We use the reliable TCP protocol at the transport layer in our study since it is widely used and supported by the operating systems of network devices. The version that we use for congestion avoidance is TCP Reno since it has a stable implementation in OMNeT++. We employed the OLSR [30] routing protocol since we assume that the

MRs are aware of the network topology. Furthermore, it scales well for few hundreds of nodes [31]. We used the hop-count as the link cost metric since the *KRR* schemes we are comparing with use the hop-count metric in the cost function. At the MAC/PHY layer, MRs are equipped with IEEE 802.11g radio functioning at 18 Mbps link rate and a transmission range of 140 meters. The access link for a MC operates on 11 Mbps. The carrier frequency is set to 2.4 GHz, maximum transmission power = 20 dBm, the noise level = -110 dBm and the Rayleigh channel model is used.

4.3 Results and Discussions

To evaluate the performance of our scheme, we have implemented the four heuristics presented in *KRR* and the *SP-DNA* heuristic. The performance metrics used for comparison are throughput, hop-count, convergence time, and communication overhead. Each of them is defined in an individual subsection. Since the obtained results in each plot point are close to the average value, we omit the confidence intervals to avoid overlapping between the error bars, which make it hard to read the graphs. However, Table 4.6 on page 94 gives the reader an idea about the range of the confidence interval for each heuristic and each performance metric.

4.3.1 Average Throughput

This is the summation of all the sizes of served objects divided by the total time taken to complete the transmission of all the packets. In this experiment set, we investigate the throughput performance by varying: (*i*) Number of distinct objects ($\mathcal{M}$); (*ii*) Zipfian parameter (α); and (*iii*) Replication period (τ).

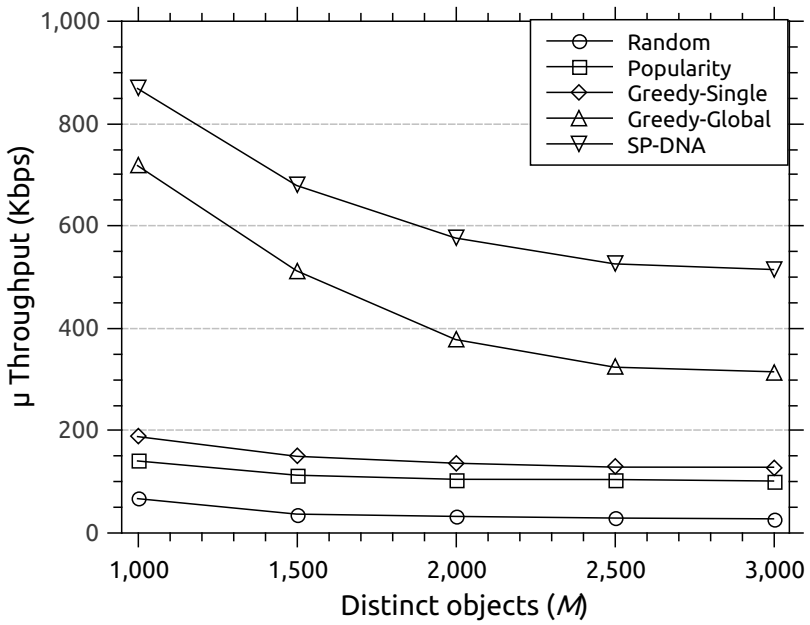


Figure 4.3: Average throughput vs. number of distinct objects $\mathcal{M}$.

Distinct objects ($\mathcal{M}$)

We compare the performance of *SP-DNA* and *KRR* by varying $\mathcal{M}$. The number of nodes $\mathcal{N} = 100$ and $\tau = 90$ minutes. The results in Fig. 4.3 show that *SP-DNA* has a significant performance gain over the basic heuristics for different $\mathcal{M}$. This is because the basic heuristics do not have any form of cooperation that results inefficient placement (e.g., neighboring nodes might have the same object replica). In comparison with *Greedy-Global*, *SP-DNA* has a gain of 21%, 33%, 52%, 62% and 63% for $\mathcal{M}$ = 1000, 1500, 2000, 2500 and 3000 respectively. Part of the gain is due to the short convergence time of *SP-DNA* (see Fig. 4.7a) during which, requests cannot be served from the best replica server as the new placement is executed. Another reason is that, given the same popularity, *SP-DNA* favors large objects over small ones. This allows the storage to accommodate large popular objects that yields to the reduction of fetching *expensive* objects. Overall,

the throughput performance decreases as $\mathcal{M}$ increases. Increasing $\mathcal{M}$ implies less storage availability for replicas (i.e., less replicas per object) resulting in the degraded throughput. Table 4.3 shows the throughput gain percentage for each heuristic over the baseline *Random* heuristic.

Table 4.3: Throughput gain for all the heuristics over the baseline *Random* heuristic when varying the number of distinct objects $\mathcal{M}$.

$\mathcal{M}$	Popularity	Greedy-Single	Greedy-Global	SP-DNA
1000	110%	181%	969%	1193%
1500	206%	306%	1288%	1740%
2000	224%	321%	1066%	1678%
2500	258%	341%	1010%	1701%
3000	267%	364%	1038%	1760%

Zipfian parameter (α)

The zipfian parameter α is a positive real number that determines the rate of the distribution's tail decay. It has been observed that Web content request follows a Zipf-like distribution [7, 97, 99]. More specifically, the Zipf distribution is defined by a probability p_i of observing the i^{th} ranked element of an infinite sequence of objects in a single random draw from that sequence, where $p_i \propto \frac{1}{i^\alpha}$. The α parameter reflects the tendency of requesting the highly ranked objects. A large value for α means a small portion of objects are highly popular, while a small α means the distribution of the popularity between the objects is flat. The parameters' values for $\mathcal{M} = 1000$, $\mathcal{N} = 100$ and $\tau = 60$.

Fig. 4.4 shows that both *Greedy-Global* and *SP-DNA* outperform the basic heuristics with different values of α. However, the performance gain increases

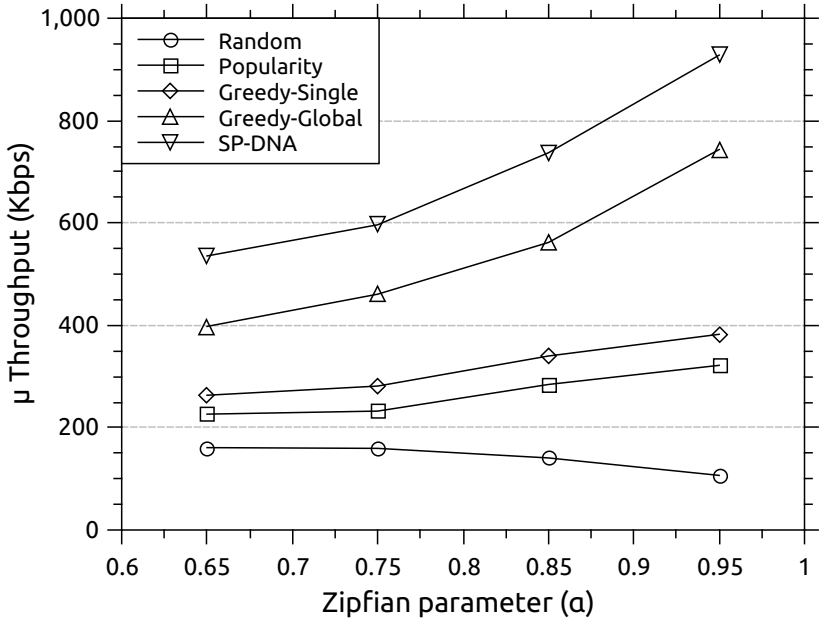


Figure 4.4: Average throughput vs. variable values of the Zipf-like skewing parameter α.

with α. This is because both *Greedy-Global* and *SP-DNA* are in favor of popular objects, creating more replicas for the popular objects and the workload request pattern will favor the highly popular objects. However, as α decreases, the request pattern changes from Zipfian distribution to Normal distribution leading to the convergence of throughput between the different schemes. Table 4.4 shows the throughput gain percentage for each heuristic over the baseline *Random* heuristic. In comparison with *Greedy-Global*, *SP-DNA* has a gain of 35%, 29%, 31% and 25% for α = 0.65, 0.75, 0.85 and 0.95 respectively. The performance gain of *SP-DNA* over *Greedy-Global* is due to:

a. The *SP-DNA* scheme is fairer towards the small objects when computing the number of replicas. This is illustrated in Eq. (4.2) that divides the popularity by the size. Therefore, the number of replicas is inversely proportional to the object size. As a result, our scheme produces a different

Table 4.4: Throughput gain for all the heuristics over the baseline *Random* heuristic when varying the Zipf-like skewing parameter α.

α	Popularity	Greedy-Single	Greedy-Global	SP-DNA
0.65	41%	64%	147%	233%
0.75	46%	76%	189%	274%
0.85	102%	141%	299%	424%
0.95	202%	258%	597%	770%

number of replicas compared to *Greedy-Global*.

b. Given the same popularity, *SP-DNA* favors large objects over small ones and as a result, minimizing the overall access cost.

c. The low convergence time (as we show in subsection 4.3.3) for *SP-DNA*. Since *Greedy-Global* requires longer time to converge than *SP-DNA* and recall that the heuristics are evaluated in online fashion, clients' requests cannot be satisfied from nearby replica servers until the placement heuristic is complete.

Replication period (τ)

This is the time period during which, object requests are observed by every node and before processing the new placement. We investigate the effect of τ's length on the throughput performance. The parameters' values for $\mathcal{M} = 1000$, $\mathcal{N} = 100$ and $\alpha = 0.95$. The results in Fig. 4.5 show that for the schemes using popularity as a factor, the throughput increases initially with τ and then starts to drop. This is due to the accuracy of the estimated popularity, which increases initially with τ as more statistical information becomes available. However, the accuracy

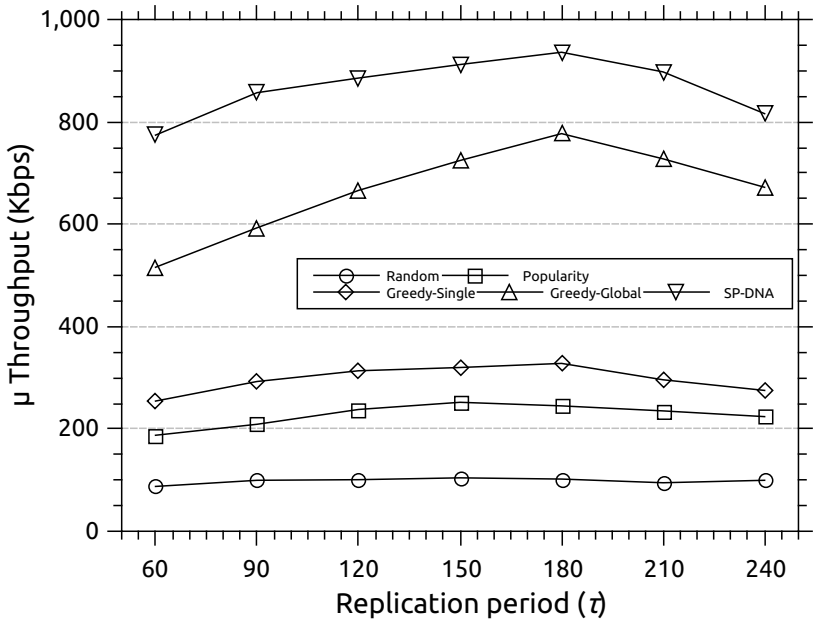


Figure 4.5: Average throughput vs. different replication periods τ.

Table 4.5: Throughput gain for all the heuristics over the baseline *Random* heuristic when varying the replication period τ.

τ	Popularity	Greedy-Single	Greedy-Global	SP-DNA
60	114%	190%	489%	784%
90	110%	194%	495%	760%
120	136%	211%	561%	779%
150	141%	207%	594%	774%
180	140%	222%	662%	818%
210	148%	212%	668%	847%
240	125%	176%	575%	719%

starts to drop as τ increases further, due to the timely nature of the popularity. Compared to *Greedy-Global*, *SP-DNA* is less affected by τ. We can also notice that the difference between *Greedy-Global* and *SP-DNA* is becomes relatively smaller with a large τ due to the reduced effect of convergence time. Table 4.5 shows the throughput gain percentage for each heuristic over the baseline *Random* heuristic.

4.3.2 Average Hop-count

This is the total number of hops between the requesting nodes and the serving nodes divided by the total number of served requests. We computed the average hop-count for all the heuristics. The $\mathcal{SC}$ is set to 512 MB, $\mathcal{M} = 1000$, $\mathcal{N} = 100$ and $\tau = 60$ minutes. The results are depicted in Fig. 4.6. It clearly shows

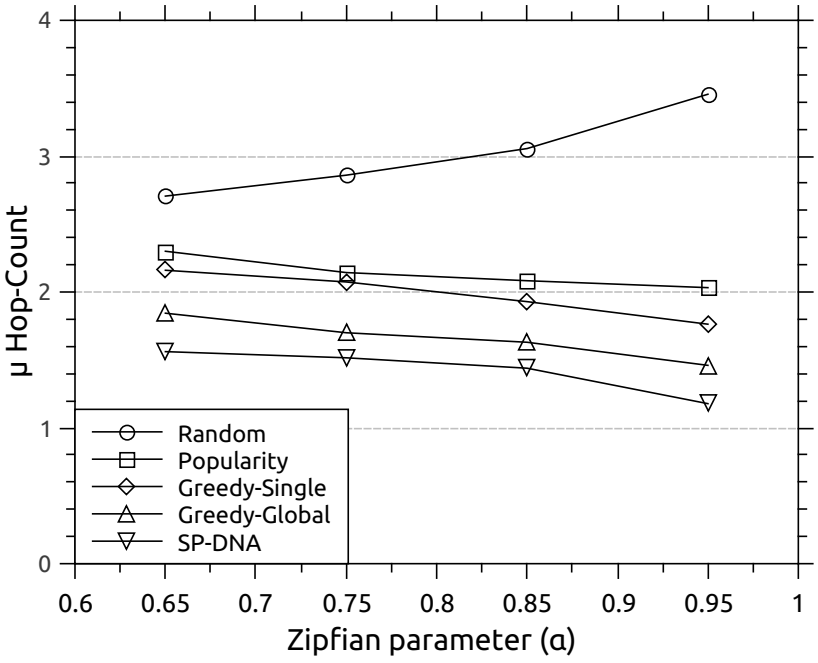
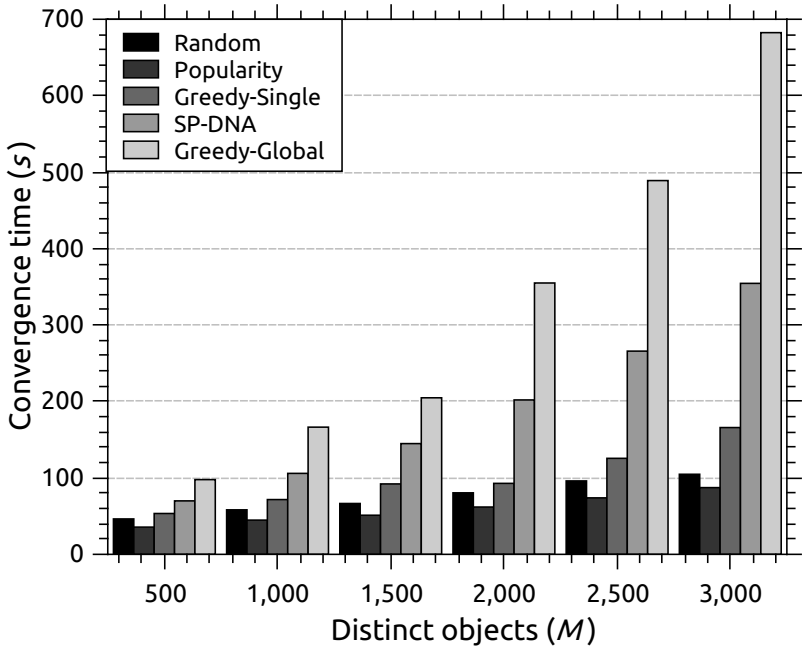


Figure 4.6: Average Hop-count vs. variable values of the Zipf-like skewing parameter α.

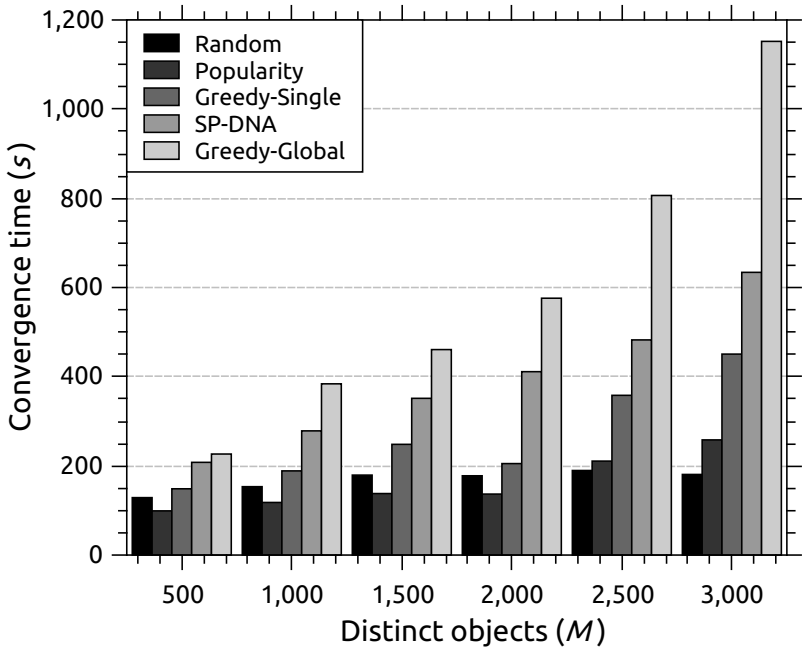
that the cooperative heuristics outperform the basic heuristics. We note that *SP-DNA* has smaller hop-count than *Greedy-Global*. This means that *SP-DNA* serves more requests over short paths. In other words, *SP-DNA* places replicas closer to the requesting nodes. The figure also shows that as α decreases the performance of the heuristics converges. As mentioned earlier, with the decrease of α the popularity difference between the objects becomes less clear. The request behavior tends to be arbitrary instead of favoring popular objects. This results in the convergence between the schemes.

4.3.3 Convergence Time

This is the total time taken to identify the number of replicas per object, decide on allocating an object replica to a node and fetching all the objects' replicas on all participating nodes for the next τ period measured in seconds. We compare the heuristics for medium ($\mathcal{N} = 100$) and large ($\mathcal{N} = 300$) network size and for different numbers of distinct objects ($\mathcal{M}$) varying between 500 and 3000. Fig. 4.7 shows that the basic heuristics (i.e., *Random*, *Popularity* and *Greedy-Single*) converge faster than their counterparts since the placement decision is taken locally without any form of cooperation. *Greedy-Global* takes the longest time since it recalculates the cost matrix after every replica placement. Recall that *Greedy-Global* performs a costly sort operation to find out the *Object-Server* pair that minimizes the access cost. Therefore, increasing $\mathcal{M}$ and/or $\mathcal{N}$, exponentially increases the convergence time of *Greedy-Global*. In contrast, since *SP-DNA* is distributed, the placement decision is performed in parallel by the DNs, therefore, runs in a polynomial time as $\mathcal{M}$ and/or $\mathcal{N}$ increase. A DN locates a replica only within its partition, and as the number of replicas increases the partition size ($\mathcal{N}'$) decreases, yielding a shorter convergence time and lower computation cost. The results reveal that *SP-DNA* significantly outperforms *Greedy-Global*.



(a)



(b)

Figure 4.7: Convergence time comparison between *KRR* and *SP-DNA* heuristics. (a) $\mathcal{N} = 100$, and (b) $\mathcal{N} = 300$.

4.3.4 Communication Overhead

This is the total number of control messages (wireless transmissions) generated from collecting popularity information and disseminating the placement decision on each node. However, this does not include the control messages of the Chord protocol. Three network sizes ($\mathcal{N} = 50$, 150 and 300) were considered to quantify the scalability of each scheme. In the simulation runs, we considered placing the replica placement entity for *Greedy-Global* in a centroid node to reduce the wireless transmissions. The results in Fig. 4.8 reveal that *Greedy-Global* incurs a high overhead that increases dramatically as the network size increases. This is caused by the hop-by-hop forwarding of popularity messages to the central entity, which finds the set of object replicas for each node and then forwards to each node its own replica set. In our hierarchical scheme, the agglomeration approach

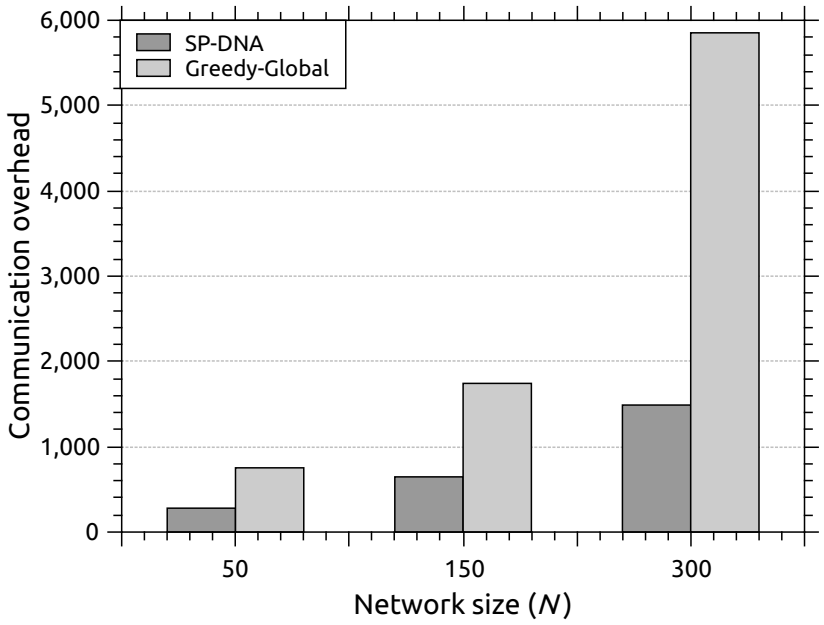


Figure 4.8: Comparison between *SP-DNA* and *Greedy-Global* heuristic in terms of communication overhead.

used by *SP-DNA* significantly reduces the message overhead since the partition members are in the proximity of their DN; the wireless transmissions traverse a short distance. As we go up the hierarchy, the agglomerated message might travel a longer distance, however, it combines multiple popularity messages from its descendants. On the other direction, when the CM assigns the placement job, it batches multiple object IDs in a multicast message (Alg. 3) that is forwarded to the corresponding DNs. Reducing communication overhead is a very important aspect in a wireless environment, where messages are highly prone to loss. We did not compare with the basic schemes, as they do not have any form of cooperation. However, they do not perform well in reducing the access cost, which is the main objective.

Table 4.6: The 95% confidence interval range for each heuristic observed in each given figure.

		Random	Popularity	Greedy-Single	Greedy-Global	SP-DNA
Fig. 4.3	Min	±2.24	±7.56	±11.76	±30.10	±49.07
	Max	±5.90	±10.10	±18.10	±54.80	±66.70
Fig. 4.4	Min	±8.12	±16.38	±22.61	±41.79	±48.86
	Max	±15.50	±25.20	±35.40	±60.90	±59.00
Fig. 4.5	Min	±6.23	±16.38	±22.89	±38.29	±56.98
	Max	±9.00	±23.50	±31.20	±75.30	±90.60
Fig. 4.6	Min	±0.14	±0.14	±0.10	±0.07	±0.07
	Max	±0.20	±0.20	±0.14	±0.10	±0.10

4.4 Summary

In this chapter, we have described our object replication and placement scheme for WMNs. To make our scheme distributed and hierarchical, we firstly build a balanced binary tree of multi-level partitions of the network. This is then used to facilitate replica placement and reduce computation and communication cost. The scheme makes the placement decision in a hierarchical way such that for a given number of replicas for a subset of the objects, a corresponding set of delegate nodes will perform the placement for the received batch of objects. This approach downsizes the problem from p-median into 1-median. The scheme takes into account the factors of object popularity and size to compute the number of replicas per object. Moreover, the scheme converges fast due to its distributed nature. Our simulation results show that the proposed replication scheme can significantly improve network performance in terms of throughput, hop-count, convergence time, and communication overhead.

Chapter 5

A Flat Approach for Object Replication

In the previous chapter, the generated partitions are based on bipartiting the graph (or the partition). This results unequal partitions' sizes when the number of replicas is not equal to 2^l, where l represents a particular level in the Delegate Binary Tree (DBT). This might yield to improper placement of object replicas that affects the access cost. In this chapter, we propose a new replication scheme and a placement heuristic called *MP-DNA* (Multiple Partitions per Delegate Node Assignment). *MP-DNA* is a distributed and scalable object replica placement scheme that uses graph partitioning to assign a delegate node for each partition. Contrary to the hierarchical *SP-DNA*, *MP-DNA* uses a flat approach to generate (semi) equal-sized partitions. We anticipate that adopting this approach can enhance the placement of replicas since the established replicas can be assigned to equal sized partitions. In this scenario, we assume that the popularity of objects is uniformly distributed across all the nodes. This does not mean that an object has exactly the same popularity or rank over all nodes, however the popularity difference is small. This assumption is valid, because Web content

follows a Zipf-like distribution in which a small portion of the objects is highly popular and the rest of the objects are lowly popular (heavy-tailed distribution).

5.1 Our Proposed MP-DNA Scheme

In this section, we present our proposed scheme. Since MRs are limited with resources and because the wireless medium is prone to packet loss, therefore, the replica placement decision should not rely on a central entity. Our proposed scheme is focused on decomposing the replica placement problem by using graph partitioning techniques. The main goal of graph partitioning is to divide a graph into a set of sub-graphs such that each sub-graph has roughly the same number of nodes and the sum of all edges that connect different sub-graphs is minimized. Therefore, graph partitioning is useful to distribute the problem into a set of partially independent sub-problems especially when dealing with NP-Complete problems. The search space is split according to the computed number of replicas per object. In each partition, a predetermined *Delegate Node* (DN) solves part of the complete search space. To this end, the replica placement problem is divided into a set of smaller loosely connected ones. Thus, we can take advantage of graph partitioning techniques to solve the problem considered in a divide-and-conquer approach.

To distribute the replica placement problem, we simplify the p-$median$ problem by partitioning the network graph into p sub-graphs, where p represents a potential number of replicas. Then we select for each partition a DN, which will be responsible for placing an object replica within its partition. The partitioning algorithm we use is proposed in [91] and has been implemented by Metis [92] software. *MP-DNA* involves two phases, the *Network Setup Phase* and the *Content Replication and Placement Phase*. In the *MP-DNA* scheme, for a given number

Algorithm 5: Pseudo code to assign DNs to partitions performed by the CM.

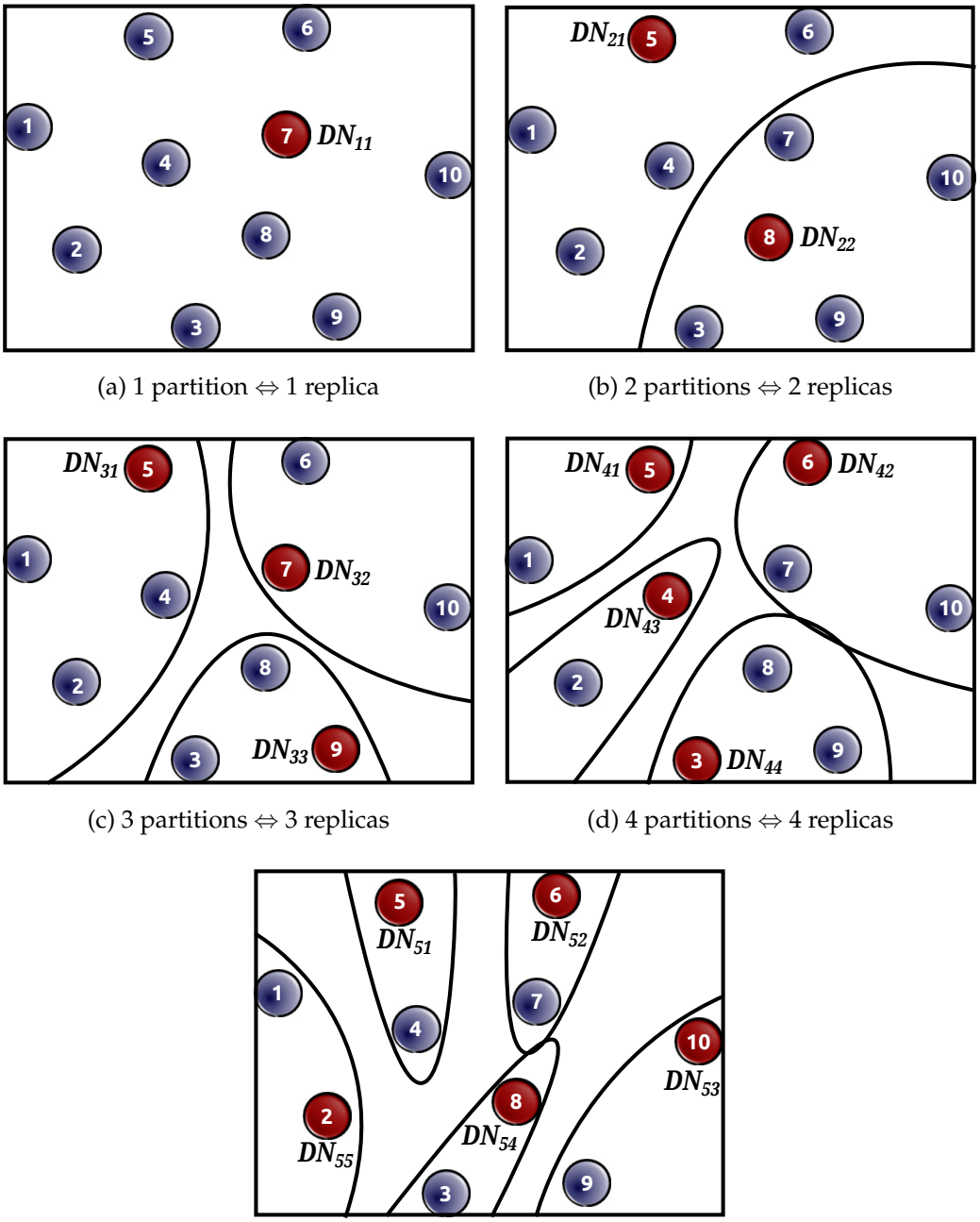
```
for u ← 1 to 𝒩 do
    Partition (G, u)
    for v ← 1 to u do
        if |G(u, v)| = 1 then
            delegate(u, v) ← G(u, v)
        else
            delegate(u, v) ← pick a node from partition G(u, v) with the least
            number of assigned partitions
        end
        Map(delegate(u, v), G(u, v)) → u in Map-List
    end
end
```

of replicas per object, the partitions are known *a priori* according to Alg. 5 such that given p replicas, there would be p corresponding partitions. It then assigns a *Delegate Node* (DN) for each partition. The DN is responsible for the placement of a single object replica out of p in its partition for each object ID it receives. Our scheme involves two phases, namely the *Network Setup Phase* and the *Content Replication and Placement Phase*.

5.1.1 Network Setup Phase

In this phase, an application-level process called the *Content Manager* (CM) runs Alg. 5. The role of the CM is to generate equal-sized partitions to match a potential number of replicas. Theoretically, the number of replicas can range from 0 to $\mathcal{N}$. The CM also assigns a Delegate Node (DN) for each partition and receives statistical information from the DNs about content popularity. We note that the CM can be hosted in a centroid MR to be close to all mesh nodes. Fig. 5.1 illustrates the generation of partitions for potential number of replicas and the assignment of a DN for each partition.

When the partition size is equal to 1 (line 4), the node forming the partition is flagged as a *self-hosting* DN. Otherwise; the CM selects a DN from the partition members with the least number of assigned partitions, and flags it as a *partition-hosting* DN. The reason why we select a member with the least number of assigned partitions is to balance the job assignment between the nodes. The CM maps the DN and its partition members to u and inserts the entry in a structure called Map-List. When Alg. 5 terminates, the CM broadcasts the resulting Map-List to all the MRs so that a DN knows which partition(s) it is responsible for. We note that this phase needs to be performed at startup or if the topology changes permanently (e.g., relocation of MRs).



(a) 1 partition ⇔ 1 replica

(b) 2 partitions ⇔ 2 replicas

(c) 3 partitions ⇔ 3 replicas

(d) 4 partitions ⇔ 4 replicas

(e) 5 partitions ⇔ 5 replicas

Figure 5.1: This figure depicts the first 5 stages of the Network Setup Phase, where each stage represents the generation of (semi) equal partition sizes and assigning a DN for each partition.

5.1.2 Content Replication and Placement Phase

This phase consists of three steps to be performed aiming to collect statistical information about the objects' request count, compute the number of replicas per object and then find their placement. These steps are described as follows:

During the τ period

Every node maintains a list containing the 4-tuple $\langle MR_i, \mathcal{O}_m, |\mathcal{O}_m|, \lambda_{mi}(\tau)\rangle$ representing the request count $\lambda_{mi}(\tau)$ for every object $\mathcal{O}_m$ during the replication period τ at node i. A request is counted, if it was initiated by a mesh client associated to the mesh router MR_i.

At the end of the τ period

The DNs send the collected statistical information to the CM, which in turn computes $p_m(\tau + 1)$ for the next τ. This is described in the following sub-steps to be performed:

a. The partition member of the smallest partition size forwards its object request count list to its delegate node (Note that a node can be a member of multiple partitions; hence, the heuristic is called *MP-DNA* (Multiple Partitions per Delegate Node Assignment)). The DN aggregates, stores and forwards the received information to the first DN responsible for a larger partition such that, the current partition set of nodes $\{\mathcal{DN}_{current}\}$ must be a subset of the selected partition $\{\mathcal{DN}_{up}\}$ up in the Map-List. This approach reduces the number of messages that disseminate the popularity statistics. This is repeated by each DN until the CM (the DN responsible for the whole network) receives the full object frequency list.

   We give an example to illustrate how this is achieved. With reference to

Fig. 5.1, we start from the level where the partition size is 2 (see Fig. 5.1e). For instance, node 1 forwards its object frequency list to DN_{55}, which aggregates the received list with its own list, stores the resulting list and then forwards it to DN_{31} (Fig. 5.1c). Note that none of the partitions in Fig. 5.1d is a superset for DN_{55}'s partition, therefore, it is forwarded to DN_{31}. Then DN_{31} receives the object frequency list from node 4, aggregates, stores and forwards the resulting list to DN_{21} (Fig. 5.1b), which is the same node (node 5); hence, this is not considered as a message. Then, DN_{21} receives the object frequency list from node 6, aggregates, stores and forwards the merged list to DN_{11}.

b. The CM merges the received information, removing any redundant entries that come as a result of a node acting as a DN for multiple partitions, where partition members overlap across different partitions. Then for each $\mathcal{O}_m$, it finds the global popularity according to Eq. (5.1).

$$Pr_m(\tau) = \frac{\sum_{i=1}^{\mathcal{N}} \lambda_{mi}(\tau)}{\sum_{i=1}^{\mathcal{N}} \sum_{m=1}^{\mathcal{M}} \lambda_{mi}(\tau)} \tag{5.1}$$

c. The CM computes the number of replicas $p_m(\tau+1)$ for every $\mathcal{O}_m$ based on Eq. (5.2).

$$p_m(\tau+1) = \frac{\mathcal{SC} \times \mathcal{N} \times \mathcal{P}r_m(\tau)}{|\mathcal{O}_m|} \tag{5.2}$$

d. The CM creates a *Replica List* ($\mathcal{RL}$) that contains the 3-tuple in the form of $\langle p_m(\tau+1), \mathcal{O}_m, |\mathcal{O}_m| \rangle$ grouped in a decreasing order by $p_m(\tau+1)$ to give priority to the highly demanded objects. Then, within each group, the objects are sorted by the object size ($|\mathcal{O}_m|$) in a decreasing order to maximize the storage benefit or in other words store the *expensive* objects that incur a high cost to be retrieved.

Algorithm 6: Pseudo code for the *MP-DNA* Heuristic.

foreach *group* $\in \mathcal{RL}$ **do**
    multicast a message containing the group of object replicas to the corresponding list of DNs in Map-List
end
upon receiving the message, a DN will do:
lookup the Map-List for the given $p_m(\tau + 1)$
if *the node is flagged s* **then**
    forall the $\mathcal{O}bj_m$ *in the received group* **do**
        `Fetch(`$\mathcal{O}bj_m$`)`
    end
else
    forall the $\mathcal{O}bj_m$ *in the received group* **do**
        select the node that minimizes the cost:
        $Min \sum_{i'=1}^{\mathcal{N}'} \sum_{j'=1}^{\mathcal{N}'} Pr_{mi'}(\tau) d_{i'j'}$ $s.t \exists SC$
        assign $\mathcal{O}bj_m$ to the selected node
        `Fetch(`$\mathcal{O}bj_m$`)`
    end
end

Replica assignment and placement

In this subsection, we describe our distributed replica placement heuristic (see Alg. 6). Based on the produced $\mathcal{RL}$, the CM multicasts a message for each group in $\mathcal{RL}$ to the corresponding DNs obtained from the Map-List in the *Network Setup Phase*. The message is in the form of a 3-tuple $\langle p_m(\tau+1), \mathcal{O}_m, |\mathcal{O}_m| \rangle$ for each object in the group. Afterwards, each DN finds out from the Map-List whether it has to *self-host* or *partition-host* the objects in the received multicast message. If the node has to *self-host* the listed objects, then it only needs to fetch them by running the same fetching algorithm Alg. 4 (see page 79). Otherwise (i.e., *partition-host*), the DN computes the total cost (line 13) for every partition node and assigns $\mathcal{O}_m$ to the node that minimizes the total cost. Note that the selected node must have enough storage to accommodate $\mathcal{O}_m$. Otherwise, the second best node will be selected and so forth. Upon successful assignment the selected node fetches the object using Alg. 4.

5.2 Simulation Experiments

We evaluate the *MP-DNA* scheme using OMNeT++ simulation framework. We use the same models for the network, content, routing, transport and MAC that were described in Chapter 4 (see page 82). We simulated a stationary WMN consisting of 100 MRs in arbitrary mesh topology. The average distance between a MR and its neighbor is 100 meters. MRs are equipped with 802.11g radio functioning at 18 Mbps link bandwidth that is appropriate to achieve a transmission range of 150 meters. The access link operates on 11 Mbps bandwidth. The OLSR routing protocol is used to route the packets. Each MR is allocated to 6 active MCs on average and each MC requests an object in a period ranging between 5

Table 5.1: Default simulation parameters

Parameter	Default Value
Simulation area	800m x 800m
Number of MRs ($\mathcal{N}$)	100
Radio interface	IEEE 802.11g
Link rate	18 Mbps
Transmission range	150m
Zipf-like parameter α	0.83
Carrier frequency	2.4 GHz
Max. Tx power	20 dBm
Noise level	-110 dBm
Channel model	Rayleigh
$\mathcal{M}$	1000
$\|\mathcal{O}_m\|$ range	$1 \Rightarrow 4$ MB
τ	90 minutes
$\mathcal{SC}$	512 MB

and 10 minutes after the previous request has been completed. The object size ranges from 1 to 4 MB uniformly distributed. MCs request objects following a Zipf-like distribution with the Zipfian parameter $\alpha = 0.83$ [7]. The distance $d_{i'j'}$ considered in the simulation is the hop-count since the heuristics we are comparing with; use the hop-count in their cost function. This is beneficial to these heuristics and serves as a baseline metric for our scheme. The application traffic used is FTP since the traffic we consider is characterized by the multimedia content and not the normal Web browsing behavior. The simulation time is set to 90 minutes and $\mathcal{SC}$ = 512 MB. The objects pool size $\mathcal{M} = 1000$. We note that in reality, the system will use larger $\mathcal{SC}$ and larger object size range $\mathcal{O}_m$. However, due to the huge working set size and length of the experiments, we scaled down both parameters. The relative increase of storage and object size will still make

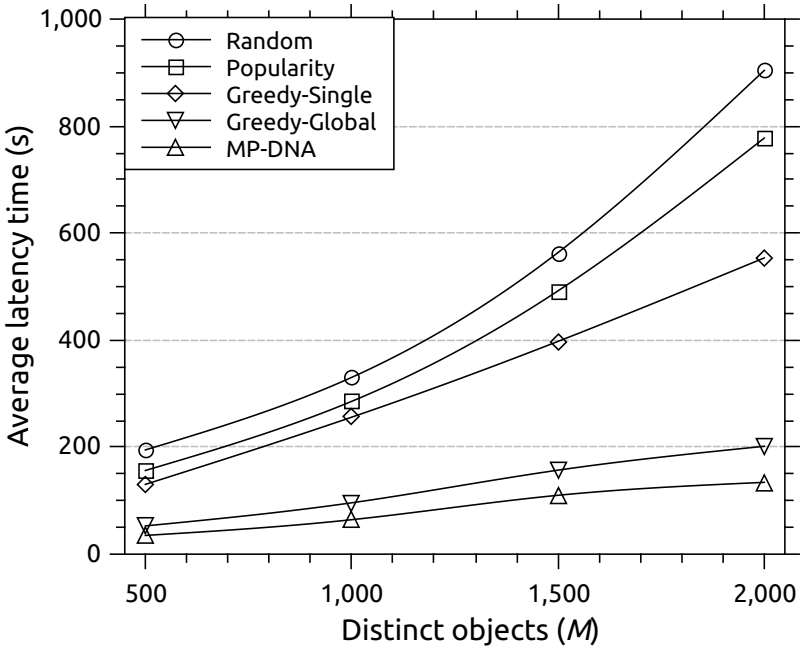
our evaluation viable as in [8]. Table 5.1 summarizes the default simulation parameters.

5.3 Results and Discussions

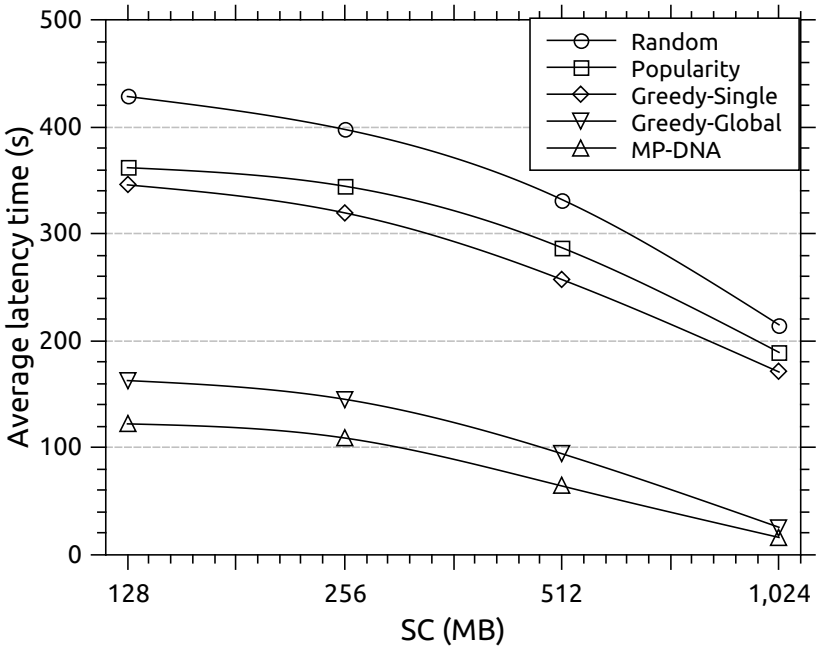
The *KRR* heuristics are used for comparison with *MP-DNA* as in Chapter 4, since the problem considered is similar to the one in *KRR*, except that their model does not consider different sizes of objects in selecting the *Node-Object* pair, while we consider it (Eq. (5.2)) in computing the number of replicas $p_m(\tau + 1)$. To evaluate the efficiency of our heuristic, we have implemented the four heuristics presented in *KRR*. We run the baseline *Random* heuristic for multiple of times and then use the same data set and generated requests to test the other *KRR* heuristics or the *MP-DNA* heuristic. The simulation results were averaged over 10 multiple random scenarios. The simulation tests are carried out in a transient state system and online fashion such that object replica placement is performed whilst clients' requests are generated. For all the heuristics, requests are served from the closest replica server. Since the obtained results in each plot point are close to the average value, we omit the confidence intervals to avoid overlapping between the error bars, which make it hard to read the graphs. However, Table 5.4 on page 117 gives the reader an idea about the range of the confidence interval for each heuristic and each performance metric.

5.3.1 Average Latency Time

In this subsection, we compare the performance of *MP-DNA* and KRR with respect to the average latency time. This is the system-wide summation of the time taken to serve all clients' requests (the time elapsed since a content request is issued until being completely downloaded) divided by the total number of



(a) $\mathcal{SC} = 512$ (MB)



(b) $\mathcal{M} = 1000$ objects

Figure 5.2: Average Latency Time.

served requests measured in seconds. In one scenario, we ran the simulation for a varying number of $\mathcal{M}$ objects with $\mathcal{SC}$ set to 512 MB. In another scenario, we evaluated the performance by varying $\mathcal{SC}$ with $\mathcal{M}$ set to 1000. For both scenarios the network size $\mathcal{N} = 100$. Fig. 5.2 (a and b) depicts the comparison for the two scenarios. Fig. 5.2a clearly shows that *MP-DNA* and *Greedy-Global* outperform the three basic heuristics. The performance gap increases with $\mathcal{M}$. This is because when $\mathcal{M}$ is small, the nodes have sufficient storage to accommodate a large number of replicas. However, as $\mathcal{M}$ increases, for the three basic heuristics the lack of placement cooperation takes its toll. Recall that for the three basic heuristics, the placement decision is taken locally without considering whether an identical replica is already placed in a nearby node. This prevents discovering cases where some objects might have a better gain if replicated in other node(s). *MP-DNA* has a significant performance gain over the basic heuristics. This is due to the lack of cooperation in the basic heuristics. We note that *MP-DNA* performs better than *Greedy-Global* when $\mathcal{M}$ increases. This gain is because *MP-DNA* considers the object size (Eq. (5.2)) in finding the density share for each object. If we fix all the variables in Eq. (5.2) and vary the object size, it yields that smaller objects have more replicas than bigger ones. In contrast, *Greedy-Global* is oblivious of the object size. Thus it is not fair towards the relatively smaller objects. This may not be an issue if we have abundant storage or the number of objects is small. However, as $\mathcal{M}$ increases, the smaller objects suffer more from the storage constraint since storing a large object could be at the cost of storing multiple small ones. Table 5.2 shows the latency gain for each heuristic over the baseline *Random* heuristic.

Fig. 5.2b shows that for different $\mathcal{SC}$, *MP-DNA* has a significant performance gain over the basic heuristics. The gain is due to the efficient replica placement that operates in a cooperative way and thus, minimizes the object access cost.

Table 5.2: Latency gain for all the heuristics over the baseline *Random* heuristic when varying the number of distinct objects $\mathcal{M}$.

$\mathcal{M}$	Popularity	Greedy-Single	Greedy-Global	MP-DNA
500	20%	33%	73%	82%
1000	14%	23%	71%	81%
1500	13%	29%	72%	80%
2000	14%	39%	78%	85%

Moreover, our placement adapts to the change in clients' demands over time and adjusts the density of replicas according to their popularity. *MP-DNA* outperforms *Greedy-Global* when $\mathcal{SC}$ is relatively small. This is due to the efficient utilization of the storage, which on one hand is fair to the small objects when computing the number of replicas (Eq. (5.2)). On the other hand, it prioritizes the large objects that are popular as the cost of fetching large popular objects is high. Another factor for the gain is due to the high convergence time of *Greedy-Global* caused by the exhaustive cost recalculations and sorting operations after every replica placement. During the convergence time, the access cost of content is not predictable, which further deteriorates user experience due to long convergence time. Table 5.3 shows the latency gain for each heuristic over the baseline *Random* heuristic.

5.3.2 Average Throughput

In this subsection, we investigate the effect of varying the replication period τ on the throughput performance. The throughput is the summation of all the sizes of served objects divided by the total time taken to complete the transmission of all the packets. The $\mathcal{SC}$ is set to 512 MB, $\mathcal{M} = 3000$, $\mathcal{N} = 100$ and $\alpha = 0.83$.

Table 5.3: Latency gain for all the heuristics over the baseline *Random* heuristic when varying the storage capacity $\mathcal{SC}$.

$\mathcal{SC}$	Popularity	Greedy-Single	Greedy-Global	MP-DNA
128	16%	19%	62%	71%
256	13%	20%	63%	73%
512	14%	23%	72%	81%
1024	12%	21%	88%	92%

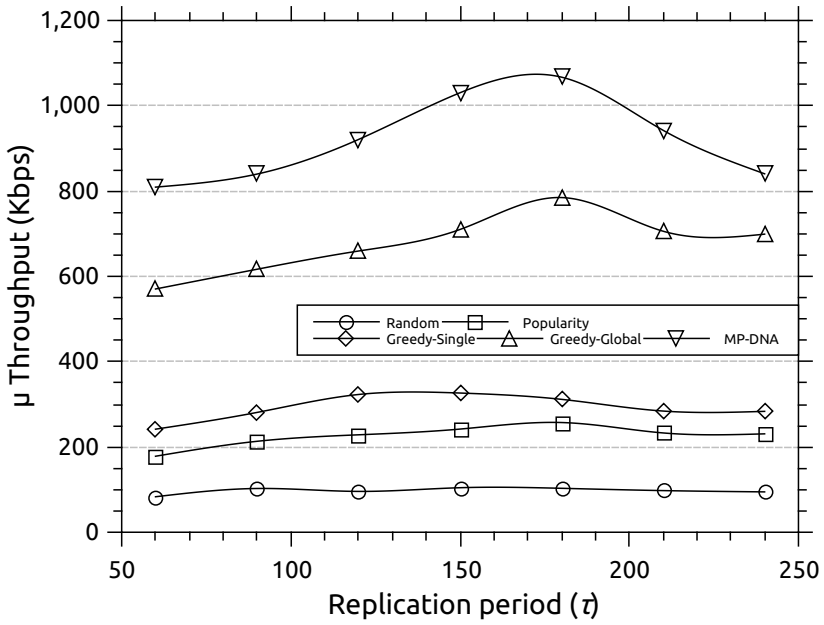
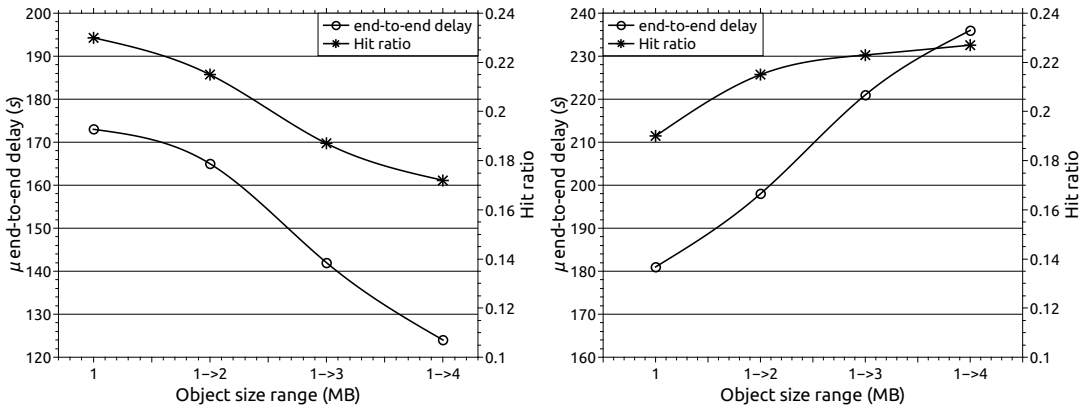


Figure 5.3: Average throughput vs. different values of τ.

The results in Fig. 5.3 show that, for the schemes using popularity as a factor, the throughput increases initially with τ and then starts to drop. This is due to the accuracy of the estimated popularity. The accuracy increases initially with τ, due to the collection of more accurate statistical information becoming available. However, the accuracy starts to drop as τ increases further, due to the timely nature of the popularity. We can notice that the baseline *Random* heuristic performs worst since it does not use the popularity statistics in the placement decision. Compared to *Greedy-Global*, the performance gain for *MP-DNA* ranges between 20% and 47%. This gain is due to the Byte-level fairness (see Eq. (5.2)) in the *MP-DNA* scheme, which yields to different object density compared with the *Greedy-Global*. Recall that *MP-DNA* favors the large popular objects than the small ones. Since we have a storage constraint on $\mathcal{SC}$ and as the pool of distinct objects ($\mathcal{M}$) increases, the number of object replicas will decrease for both *Greedy-Global* and *MP-DNA*. Another reason that explains the gain is that as $\mathcal{M}$ increases, the convergence time for *Greedy-Global* increases significantly; hence, objects cannot be solicited from the nearest server during the convergence time.

5.3.3 Average Latency Time vs. Hit Ratio

In these simulation tests, we investigate the effectiveness of sorting objects of semi-equal popularity based on objects' size both in decreasing and increasing orders. This is a trade-off between latency and hit ratio as can be noticed in both Figs. 5.4a and 5.4b. Prioritizing large objects (see Fig. 5.4a) shows that as the range increases, the latency improves as the placement optimality of large objects converges, which reduces the amount of packets traversing. However, the hit ratio drops slightly as the range increases. On the contrary, Fig. 5.4b shows a latency increment as the size gap increases and an improvement in the hit ratio



(a) Decreasing order

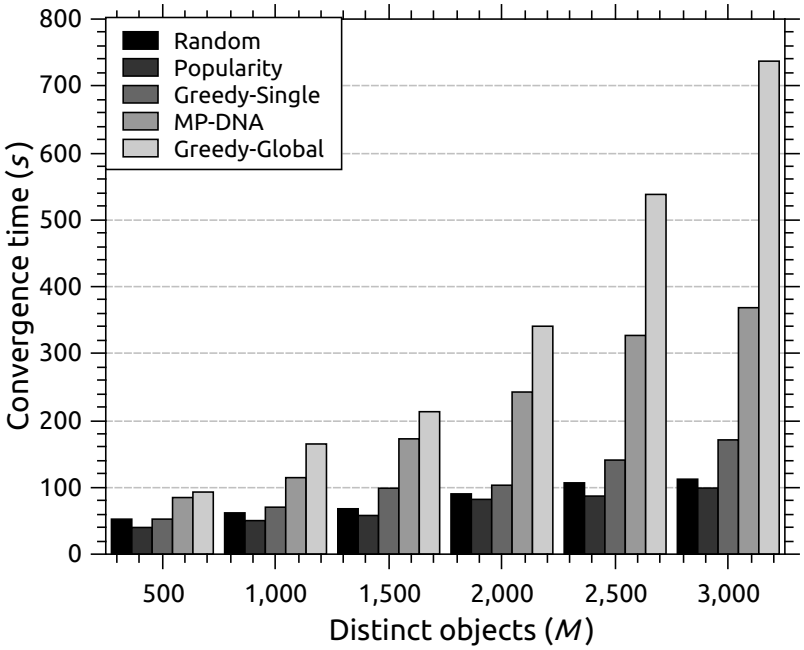
(b) Increasing order

Figure 5.4: The trade-off between the average latency time and the hit ratio vs. variable ranges of object size.

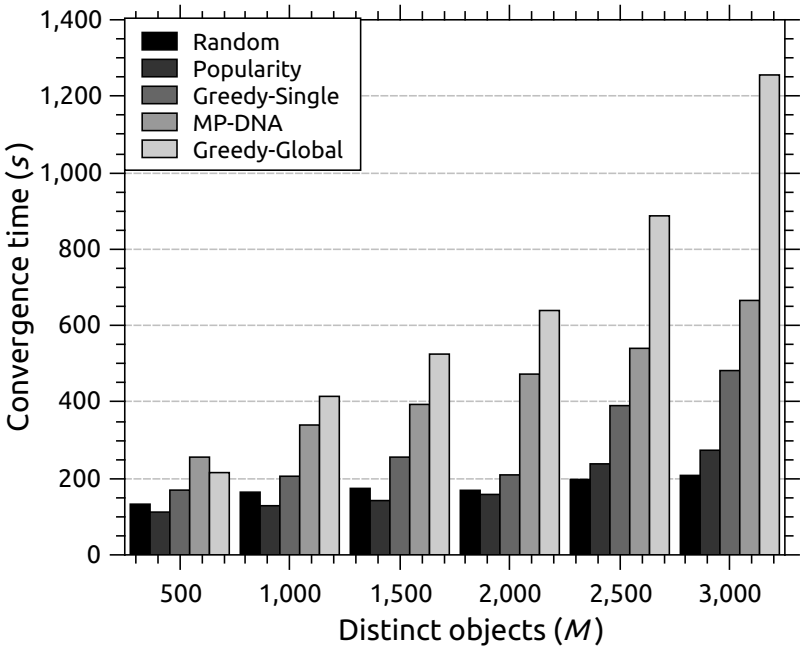
since replicating smaller objects usually results in higher hit ratios [99]. However, beyond a limit (i.e., 1⇒3) this improvement becomes slight; hence, we favor the decremental sort as the benefit in access cost is more than the benefit in the hit ratio.

5.3.4 Convergence Time

In this subsection, we compare *MP-DNA* with *KRR* from the convergence time perspective. This is the total time taken to decide on allocating all the object replicas on all participating nodes for the next τ period measured in seconds. The network sizes used were medium ($\mathcal{N} = 100$) and large ($\mathcal{N} = 300$). The number of distinct objects ($\mathcal{M}$) varies from 500 to 3000 with $\mathcal{SC}$ set to 512 MB. Since our placement heuristic is distributed, the burden of deciding on the object placement is distributed between the *DNs*. Fig. 5.5 shows the comparison between *MP-DNA* and *KRR* with different network sizes ($\mathcal{N} = 100$ and 300). It shows that the basic heuristics (i.e., *Random*, *Popularity* and *Greedy-Single*)



(a) $\mathcal{N} = 100$



(b) $\mathcal{N} = 300$

Figure 5.5: Convergence time for different network size.

converge faster than the cooperative *Greedy-Global* and *MP-DNA* because their replica placement decision is taken locally without any form of cooperation. The results show that *Greedy-Global* requires the longest convergence time. This is because it recalculates the cost after every replica placement. It also shows that, as $\mathcal{M}$ and/or $\mathcal{N}$ increase, the convergence time of *Greedy-Global* increases exponentially indicating that *Greedy-Global* does not scale well with the number of objects and network size. This is because in *Greedy-Global*, a server performs a costly sort operation to find out the *Object-Server* pair that minimizes the access cost. In contrast, *MP-DNA* shows a logarithmic scale in the convergence time as $\mathcal{M}$ and/or $\mathcal{N}$ increases. Since *MP-DNA* is distributed, a *DN* locates a replica only within its partition. As the number of replicas increases, the size of $\mathcal{N}'$ decreases yielding a shorter convergence time. Although *MP-DNA* does not perform better than the basic heuristics, it performs much better than *Greedy-Global*. However, the basic heuristics are not comparable when it comes to the reduction in the access cost, which is the main objective of replica placement.

5.3.5 Communication Overhead

In this subsection, we compare the communication overhead for *MP-DNA* and *Greedy-Global*. We do not compare with the basic heuristics since they do not cooperate when taking the placement decision. This is the total number of control messages (wireless transmissions) generated from collecting popularity information and disseminating the placement decision on each node. Three network sizes ($\mathcal{N} = 50$, 150 and 300) were considered to quantify the scalability of each scheme. In the simulation runs, we considered placing the replica placement entity for *Greedy-Global* in a centroid node to reduce the wireless transmissions. The results in Fig. 5.6 show that *Greedy-Global* incurs a high overhead that increases dramatically as the network size increases. This is

caused by the hop-by-hop forwarding of popularity messages to the central entity, which finds the set of object replicas for each node, their placement and then forwards the list of object replicas for each node. In our flat scheme, each

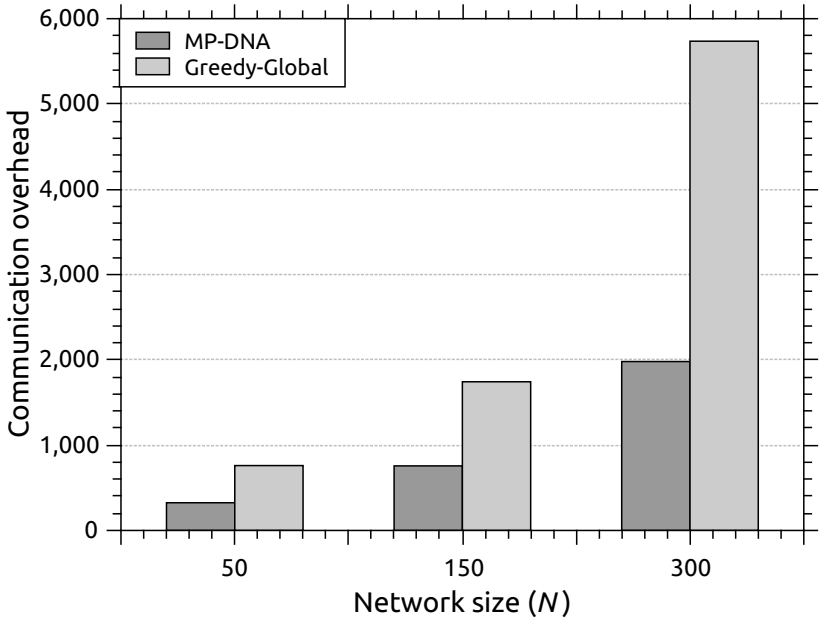


Figure 5.6: Comparison between *MP-DNA* and *Greedy-Global* heuristic in terms of communication overhead.

DN agglomerates the messages received from the DN(s) composing a subset partition. This approach significantly reduces the message overhead since the wireless transmissions traverse a short distance. As we go up the Map-List, the agglomerated message might travel a longer distance, however, it combines multiple popularity messages. On the other direction, when the CM assigns the placement job, it batches multiple object IDs in a multicast message (Alg. 6) that is forwarded to the corresponding DNs.

Table 5.4: The 95% confidence interval range for each heuristic observed in each given figure.

		Random	Popularity	Greedy-Single	Greedy-Global	MP-DNA
Fig. 5.2a	Min	±34.86	±44.87	±24.60	±5.88	±7.56
	Max	±59.10	±49.20	±31.08	±14.70	±11.90
Fig. 5.2b	Min	±28.14	±17.29	±18.00	±6.37	±6.50
	Max	±39.90	±35.00	±22.89	±12.90	±8.26
Fig. 5.3	Min	±6.37	±14.21	±22.75	±47.95	±66.57
	Max	±8.40	±23.60	±28.60	±73.10	±85.10

5.4 Summary

In this chapter, we have described and evaluated a novel content replication and placement scheme. The proposed scheme is flat, distributed, and lightweight that utilizes the storage capability of mesh routers in a distributed manner. The scheme aims to generate equal-sized partitions for different potential number of replicas. This approach can provide a better placement for the disseminated object replicas, which reduces the access latency. The scheme proves to be scalable in the sense that the p-median problem is down-sized to a 1-median problem that suits the scarce resources and accounts for the specific characteristics of WMNs. Our results show the proposed scheme can significantly improve network performance in terms of object access latency, throughput, convergence time and communication cost. However, it incurs extra computation and communication cost compared to the hierarchical *SP-DNA* scheme due to the multiple partitions' assignment for each delegate node. As a result, a delegate node will collect popularity statistics from various partitions' members yielding to extra computation/communication costs.

Chapter 6

Local Popularity Aware Object Placement

In the previous two chapters, we proposed two distributed replica placement schemes for popular Web objects over WMNs. The two schemes build an overlay P2P network formed by MRs. The two schemes exploit graph partitioning to distribute the placement problem on pre-assigned delegate nodes where each delegate node is responsible for a single partition (*SP-DNA*) or multiple partitions (*MP-DNA*). However, the two schemes were proposed with the assumption that content popularity of an object is uniformly distributed between the nodes. In a scenario where this assumption does not hold, the basic versions of the two heuristics can suffer from improper placement of object replicas in partitions that do not show a demand for such replicas.

Our contribution in this chapter is focused on considering the *local popularity* of an object replica. The local popularity can be defined as the relative demand for an object within a partition compared to the whole network. We build on our previous schemes to consider the scenario where nodes can have diverse demands for different objects. We modify the placement heuristics such that

a DN finds out whether it is *feasible* to place an object within its partition or not based on the measured local popularity and comparing it with a tunable threshold that decides on placing the object replica or forwarding it to another DN responsible for a larger partition where a better placement can be considered.

6.1 Enhancements on the Placement Heuristics

In this section, we present the modified placement heuristics for both *SP-DNA* and *MP-DNA*. The same Network Setup Phase for *SP-DNA* (page 68) and *MP-DNA* (page 100) is used to generate the different partitions in a hierarchical or flat fashion. The Content Replication and Placement Phase is also similarly used (pages 75 and 102). However, the modifications depart from sub-step d in *SP-DNA* (page 76) and sub-step d in *MP-DNA* (page 103). In the next two subsections, we highlight and describe the modifications for each scheme.

Modified Content Replication and Placement Phase

d. The CM creates a *Replica-List* ($\mathcal{RL}$) that contains information about the objects' replicas. The generated list structure is in the form of a 4-tuple $\langle p_m(\tau+1), \mathcal{O}_m, |\mathcal{O}_m|, \lambda_m(\tau)\rangle$ grouped by $p_m(\tau+1)$ in a decreasing order and within each group, the objects are sorted by their size $|\mathcal{O}_m|$ in a decreasing order. The reason behind this way of sorting is that we give priority for the highly popular objects and then the priority is given to the larger objects. Within a group of objects that have the same number of replicas, objects will have semi-equal popularity; prioritizing large objects would minimize the cost weighted by the object size given the storage constraint. However, our scheme is fair with small objects in terms of the computed number of replicas, since it divides the popularity of an object by its size.

This adopted approach of sorting was not considered in the schemes we compare our work with in this thesis and it cannot be adopted by such schemes. Therefore, they assume equal object size. In fact, our approach yields to different set of replicas and different number of replicas per object. We note here the difference in the 4-tuple, which adds a new field $\lambda_m(\tau)$. This field represents the total request count for object m throughout all participating nodes. The importance of this field is going to be discussed in the next subsection.

Replica assignment and placement

In this step, we combine the description on the modifications we made on both the *SP-DNA* heuristic (Alg. 7) and the *MP-DNA* heuristic (Alg. 8) to consider the local popularity within the partition. The CM multicasts a message that contains the 4-tuple $\langle p_m(\tau+1), \mathcal{O}_m, |\mathcal{O}_m|, \lambda_m(\tau)\rangle$ to the corresponding DNs obtained from the Map-List.

- For **SP-DNA**: When each DN receives the multicast message, a decision has to be made (line 7 in Alg. 7) to find out whether it is feasible to place $\mathcal{O}_m$ in $\mathcal{DN}_k$'s partition or forward it to $parent(\mathcal{DN}_k)$ in the Delegates Binary Tree.

- For **MP-DNA**: When each DN receives the multicast message, a decision has to be made (line 8 in Alg. 8) to find out whether it is feasible to place $\mathcal{O}_m$ in $\mathcal{DN}_k$'s partition or forward it to another DN in the Map-List (recall that the Map-List is broadcasted to all the nodes at the end of the Network Setup Phase). However, a condition that must be satisfied is: *The current partition set of nodes* $\{\mathcal{DN}_{current}\}$ *must be a subset of the selected partition* $\{\mathcal{DN}_{up}\}$ *up in the Map-List*. This is to guarantee that all the members of the

Algorithm 7: Pseudo code for the modified *SP-DNA* heuristic that considers local popularity.

foreach *group* $\in \mathcal{RL}$ **do**
    multicast a message containing the group of object replicas to the corresponding list of DNs in Map-List
end
receiving $\mathcal{DN}_k$ performs the following:
lookup the Map-List for the given $p_m(\tau + 1)$
forall the $\mathcal{O}_m$ *in the received group* **do**
    if $\epsilon_{mk} > \eta_m$ **then**
        forward $\langle p_m(\tau + 1), \mathcal{O}_m, |\mathcal{O}_m|, \lambda_m(\tau)\rangle \rightarrow parent(\mathcal{DN}_k)$
    else
        if $\mathcal{DN}_k$ *is flagged* $\mathcal{S}_{host}$ **then**
            `Fetch(`$\mathcal{O}bj_m$`)`
        else
            pick a node that minimizes the cost:

$$Min \sum_{\substack{1 \leq i' \leq \mathcal{N}' \\ 1 \leq j' \leq \mathcal{N}'}} Pr_{mi'}(\tau) d_{i'j'} s.t \exists \mathcal{SC}$$

            assign $\mathcal{O}bj_m$ to the selected node
            `Fetch(`$\mathcal{O}bj_m$`)`
        end
    end
end

Algorithm 8: Pseudo code for the modified *MP-DNA* heuristic that considers local popularity.

foreach *group* $\in \mathcal{RL}$ **do**
    multicast a message containing the group of object replicas to the corresponding list of DNs in Map-List
end
receiving $\mathcal{DN}_k$ performs the following:
upon receiving the message, a $\mathcal{DN}_k$ will:
lookup the Map-List for the given $p_m(\tau + 1)$
forall the $\mathcal{O}_m$ *in the received group* **do**
    if $\epsilon_{mk} > \eta_m$ **then**
        forward $\langle p_m(\tau + 1), \mathcal{O}_m, |\mathcal{O}_m|, \lambda_m(\tau)\rangle \rightarrow$ the first $\mathcal{DN}_{up}$ up the Map-List s.t $\{\mathcal{DN}_{current}\} \subset \{\mathcal{DN}_{up}\}$
    else
        if $\mathcal{DN}_k$ *is flagged* $\mathcal{S}_{host}$ **then**
            `Fetch(`$\mathcal{O}bj_m$`)`
        else
            pick a node that minimizes the cost:

$$Min \sum_{\substack{1 \leq i' \leq \mathcal{N}' \\ 1 \leq j' \leq \mathcal{N}'}} Pr_{mi'}(\tau) d_{i'j'} s.t \exists \mathcal{SC}$$

            assign $\mathcal{O}bj_m$ to the selected node
            `Fetch(`$\mathcal{O}bj_m$`)`
        end
    end
end

current partition are included in the search for (sub)optimal placement of the required object.

The corresponding $\mathcal{DN}_k$, computes $\mathcal{P}r_{mk}(\tau)$ using Eq. (6.1), which represents the popularity percentage of $\mathcal{O}_m$ within partition k with respect to the whole network. Then it compares the ϵ_{mk} value from Eq. (6.2) with a threshold value η_m from Eq. (6.3). ϵ_{mk} acts as a *feasibility index* to decide upon the feasibility of placing $\mathcal{O}_m$ within the DN's partition or forwarding it to $parent(\mathcal{DN}_k)$ (*SP-DNA*)/$\mathcal{DN}_{up}$ (*MP-DNA*). The threshold η_m can be tuned using the coefficient θ, to trade-off between placement accuracy and computation cost for a given partition size. If ϵ_{mk} exceeds η_m, then $\mathcal{DN}_k$ will handoff $\mathcal{O}_m$ to $parent(\mathcal{DN}_k)/\mathcal{DN}_{up}$, which is responsible for a larger partition, where a better placement can take place. Then $parent(\mathcal{DN}_k)/\mathcal{DN}_{up}$ on its turn will take the decision of placement or handoff.

We note here that η_m is inversely proportional to the tunable coefficient θ that serves as an error margin. This means that, for a given network size and number of replicas $p_m(\tau + 1)$, a smaller θ yields a larger error margin (η_m). A larger η_m means reduced communication/computation cost. However, this comes at the expense of placement accuracy. On the other hand, when θ increases, the error margin η_m decreases. This leads to a better placement for an object that is not feasible within a particular partition. Improving the placement will increase the communication/computation overhead as more object replicas will be forwarded, more control messages will be incurred and the degree of parallelism will be reduced increasing the computation cost. Note that, for large partitions, the computation cost is higher than the smaller ones. This is a trade-off between communication/computation cost and placement accuracy.

$$\mathcal{P}r_{mk}(\tau) = \frac{\sum_{i'=1}^{\mathcal{N}'} \lambda_{mi'}(\tau)}{\lambda_m(\tau)} \tag{6.1}$$

$$\epsilon_{mk} = \frac{1}{p_m(\tau+1) \times \frac{\mathcal{P}r_{mk}(\tau)}{1+\zeta_{mk}}} - 1 \tag{6.2}$$

$$\eta_m = \frac{1}{p_m(\tau+1) \times \theta} \tag{6.3}$$

If ϵ_{mk} is not larger than η_m (line 9 in Alg. 7 and line 10 in Alg. 8), it looks up the Map-List for its role. If it has to *self-host* the replica $\mathcal{O}_m$, then it only needs to fetch it (line 11 in Alg. 7 and line 12 in Alg. 8). Otherwise (i.e., *partition-host*), $\mathcal{DN}_k$ will compute the demand-weighted total cost for every partition member and assigns $\mathcal{O}_m$ to the node that minimizes the total cost. The distance $d_{i'j'}$ represents the sum of the shortest distance between i' and j' within the partition. Upon successful assignment the selected node fetches the object using Alg. 4. Note that if there is no sufficient space to accommodate $\mathcal{O}_m$, $\mathcal{DN}_k$ selects the second best node and so forth, even if all partition members' storage is full, the object will be forwarded to the $parent(\mathcal{DN}_k)/\mathcal{DN}_{up}$ up in the Map-List, which on its turn will decide on the feasibility of placing the forwarded object in its partition or not by running. Another note is that the *infeasible* objects that need to be forwarded to larger partitions in the Map-List are batched and forwarded after the current $\mathcal{DN}_k$ completes the placement of the *feasible* objects. This gives the priority for each DN to finish placing the original set of feasible objects that was multicasted by the CM and then placing the forwarded objects. This rule is important for two reasons:

a. DNs can simultaneously undertake the placement of their own *feasible* objects, which increases the degree of parallelism; and

b. Make sure that when a DN has to place forwarded objects, the placement should consider other replicas ζ_{mk} of the same object (if any) in the underlying partitions.

Numerical example for the local popularity in *SP-DNA*

We give a numerical example (see Table 6.1) to describe how the placement decision is made. We use the graph and DBT of Fig. 4.1 in page 72. Suppose an object requires 3 replicas (i.e., $p_m(\tau+1)=3$). For instance, the DN set will be nodes 3, 6 and 7 according to the Map-List. Each DN decides based on the local popularity $\mathcal{P}r_{mk}(\tau)$ within its partition. Therefore, node 3 will forward the placement job to node 2, while both nodes 6 and 7 will find the optimal placement in each partition. We note here that when node 2 receives the object placement request from node 3, the value of $\zeta_{mk}=1$. This is because node 6 already placed a replica in its partition, which is a subset of node 2's partition.

Table 6.1: Numerical example showing how a DN in *SP-DNA* decides to *place* an object in its partition or *forward* it to its parent DN given $p_m(\tau+1)=3$, $\theta=1$, $\eta_m=0.33$, and $\zeta_{mk}=0$.

$\mathcal{DN}_k$	$\mathcal{P}r_{mk}(\tau)$	ϵ_{mk}	Decision ($\epsilon_{mk}>\eta_m$)
3	10%	2.33	Handoff
6	50%	-0.33	Place
7	40%	-0.17	Place

Numerical example for the local popularity in *MP-DNA*

We give a numerical example (see Table 6.2) to describe how the placement decision is made. We use Fig. 5.1 in page 101. Suppose an object requires 3 replicas ($p_m(\tau+1)=3$). For instance, the DN set will be nodes $\mathcal{DN}_{31}$, $\mathcal{DN}_{32}$ and $\mathcal{DN}_{33}$ and that no replicas of the same object have been placed previously in the given partitions (i.e., $\zeta_{mk}=0$ for all partitions). Each DN decides based on the local popularity $\mathcal{P}r_{mk}(\tau)$ within its partition. Therefore, node $\mathcal{DN}_{33}$ will

forward the placement job to node $\mathcal{DN}_{22}$ since $\{\mathcal{DN}_{33}\} \subset \{\mathcal{DN}_{22}\}$, while both nodes $\mathcal{DN}_{31}$ and $\mathcal{DN}_{32}$ will find the optimal placement in their own partition. When $\mathcal{DN}_{22}$ will decide on the placement of the forwarded object, the value of η_m will be 0.50 and suppose that $\mathcal{P}r_{mk} = 35\%$, we analyze two cases:

- **Case 1**: If $\mathcal{DN}_{32}$ did not place the same object in one of the two overlapping nodes (i.e., $\zeta_{mk} = 0$), then $\epsilon_{mk} = -0.05$ and since it is $\leq \eta_m = 0.50$, $\mathcal{DN}_{22}$ will decide to place it in its partition.
- **Case 2**: If $\mathcal{DN}_{32}$ has placed the same object in one of the two overlapping nodes (i.e., $\zeta_{mk} = 1$), then $\epsilon_{mk} = 0.90$ and since it is $> \eta_m = 0.50$, $\mathcal{DN}_{22}$ will decide to handoff the object placement to $\mathcal{DN}_{11}$, which will find a better placement according to the whole network.

Table 6.2: Numerical example showing how a DN in *MP-DNA* decides to *place* an object in its partition or *forward* it to the first $\mathcal{DN}_{up}$ given $p_m(\tau + 1) = 3$, $\theta = 1$, and $\eta_m = 0.33$.

$\mathcal{DN}_k$	$\mathcal{P}r_{mk}(\tau)$	ϵ_{mk}	Decision ($\epsilon_{mk} > \eta_m$)
$\mathcal{DN}_{31}$	30%	0.11	Place
$\mathcal{DN}_{32}$	55%	-0.39	Place
$\mathcal{DN}_{33}$	15%	1.22	Handoff

6.2 Simulation Experiments

To evaluate our scheme, we used OMNeT++ [94] simulator. We simulated a stationary WMN for different scenarios in different mesh topologies. We

Table 6.3: Default simulation parameters

Parameter	Default Value
Simulation area	800m x 600m & 1500m x 1000m
Number of MRs ($\mathcal{N}$)	50 & 150
Radio interface	IEEE 802.11g
Link rate	18 Mbps
Transmission range	150m
Zipf-like parameter α	0.83
Carrier frequency	2.4 GHz
Max. Tx power	20 dBm
Noise level	-110 dBm
Channel model	Rayleigh
Threshold coefficient θ	1
$\mathcal{M}$	3000
$\|\mathcal{O}_m\|$ range	$1 \Rightarrow 4$ MB
τ	60 minutes
$\mathcal{SC}$	512 MB

used the common OLSR [100] routing protocol. The hop-count was used to for all the schemes under evaluation. The request pattern follows a Zipf-like distribution [7]. We use the same models for the network, content, routing, transport and MAC that were described in Chapter 4 (see page 82). The browsing behavior of the clients is simulated using a random think time period that represents the time elapsed between successive web page downloads by a MC. This period ranges between 5 and 10 minutes on average since the last requested object has been downloaded. The simulation results were averaged over multiple random scenarios. The application traffic is created using FTP traffic generator. In each simulation test, the baseline *Random* heuristic was run first, followed by one of the other heuristics. For all the heuristics, requests are served from the

closest replica server. The simulation tests are carried out in transient state such that object replica placement is performed whilst clients' requests are generated. Table 6.3 summarizes the default parameters' values used.

6.3 Results and Discussions

In this section, we present the simulation results and discuss them according to the performance metrics described in each subsection. To evaluate the efficiency of our schemes, we compare their performance with the *Random* [83], *Lat-CDN* [84] and *Greedy-Global* [83] heuristics. In each simulation test, the baseline *Random* heuristic was run first, followed by one of the other heuristics. The simulation tests are carried out in transient state such that object replica placement is performed whilst clients' requests are generated.

6.3.1 Mean Hop-count vs. Threshold Coefficient (θ)

The mean hop-count is defined as the total number of hops between the requesting nodes and the serving nodes divided by the total number of served requests. In these simulation tests, we investigate the sensitivity effect of the threshold-tuning coefficient θ on the placement optimality for both schemes on one side. On the other side, we observe the effect of θ on the communication/computation cost. Two network sizes are considered $\mathcal{N} = 100$ and 300, $\mathcal{M} = 3000$, and the replication period $\tau = 60$ minutes. The results depicted in Fig. 6.1 show that for low values of θ (< 1), the hop-count cost increases as θ decreases. This is due to the increment in the error margin η_m, which allows the placement of less popular objects within the partition. As a result, the placement optimality diverges from being optimal. We can also notice that for low values of θ, the placement converges faster and incurs less communication overhead.

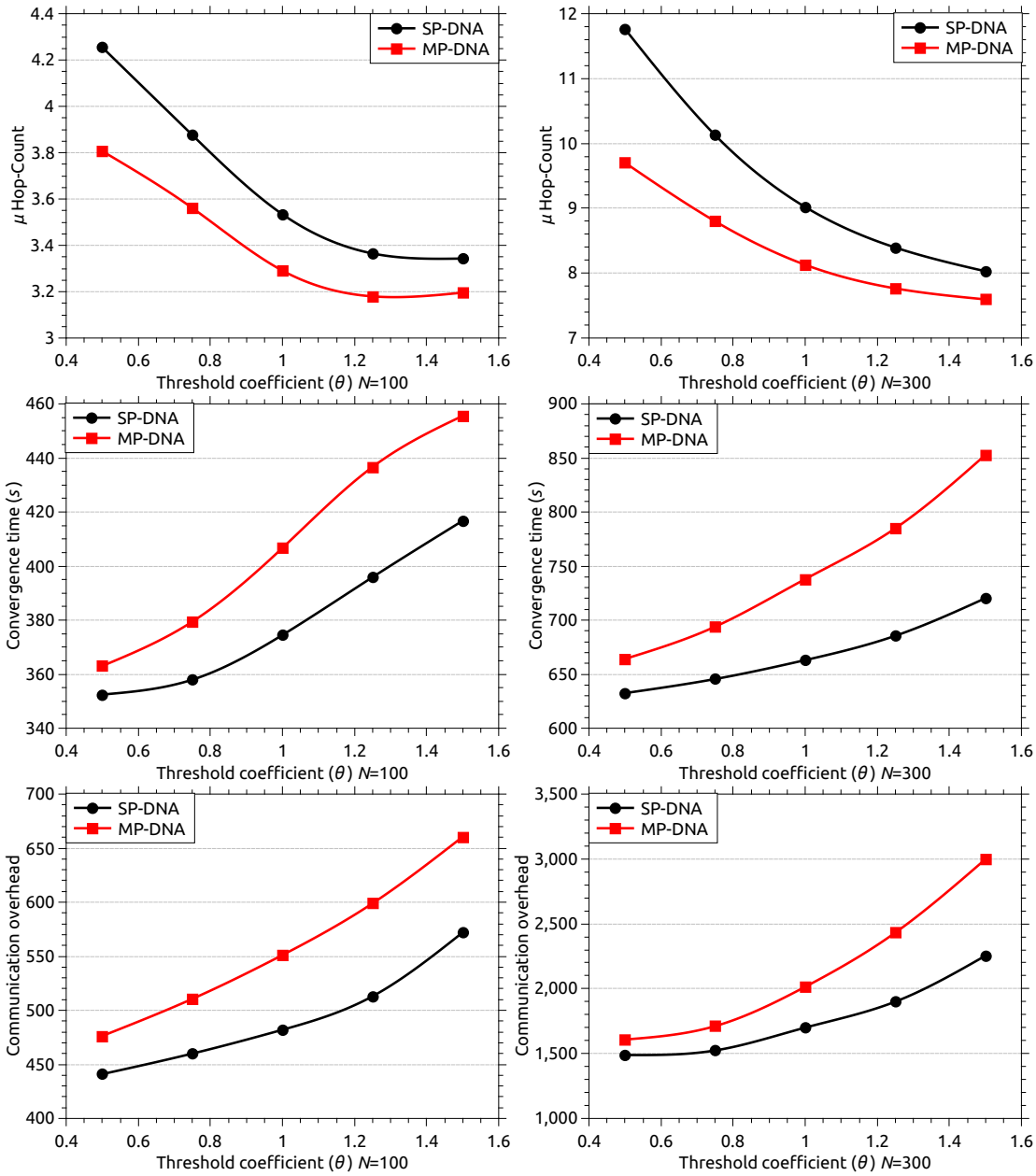


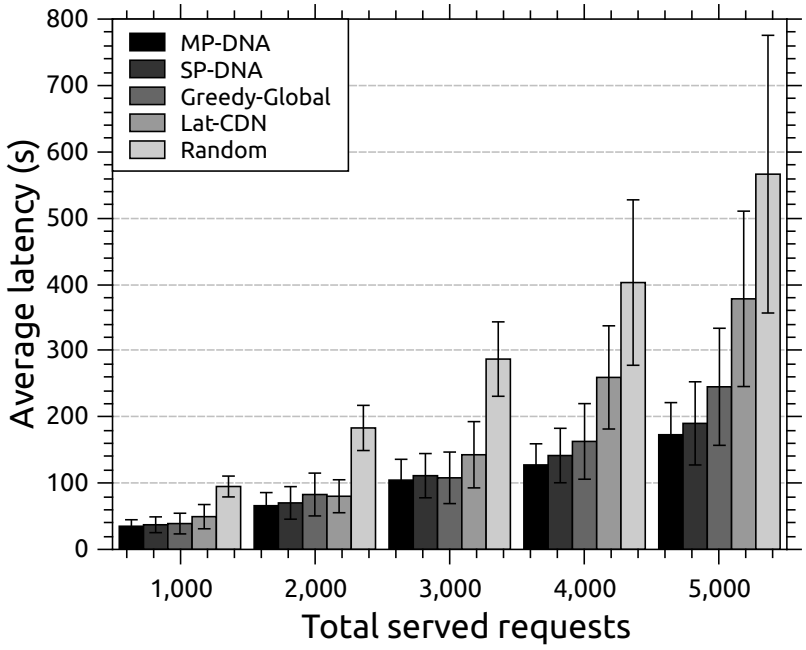
Figure 6.1: The trade-off between the mean hop-count and convergence time vs. the threshold coefficient θ. (a) $\mathcal{N} = 100$, and (b) $\mathcal{N} = 300$.

This is because the error margin η_m becomes larger and therefore, fewer objects will be forwarded to larger partitions for finding a better feasible placement. This facilitates for the placement heuristic to run in parallel and reduce the computation/communication overhead. On the other extreme of Fig. 6.1, we notice that as θ increases, the placement accuracy improves as the error margin decreases η_m. However, there is a trade-off between placement optimality and computation/communication cost. We also notice that the placement improves well to a specific limit (i.e., θ=1.25). Beyond this limit, the improvement becomes slight as the placement approaches the near optimal placement.

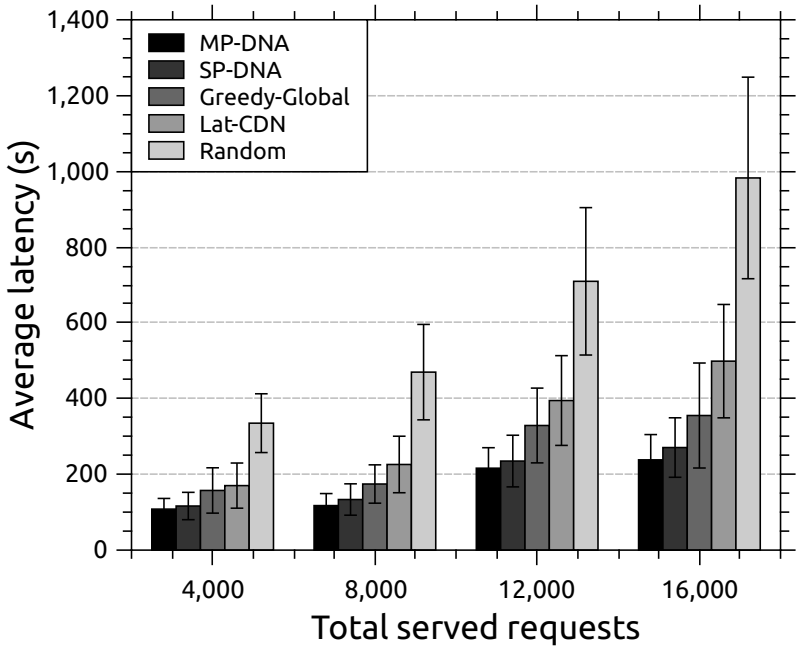
6.3.2 Average Latency Time vs. Variable Request Rate

In this subsection, we compare the performance of both *SP-DNA/MP-DNA* and the other heuristics with respect to the average latency time. In this experiment set, we compare the performance of each heuristic by varying the number of served requests within a fixed replication period τ, Two different network sizes considered, namely $\mathcal{N} = 50$ and 150 and the number of distinct objects $\mathcal{M} = 1000$ and 3000 respectively. Object size ranges between 1 and 4 MB [101] in a uniform distribution. Both figures 6.2a and 6.2b depict a comparison for the two scenarios. It can be noticed that the latency time increases with the number of served requests for all heuristics. This is because τ is fixed, and therefore, the request rate increases, which aggravates the channel competition between the upstream and downstream traffic yielding packet delay, loss, and retransmission due to channel congestion and degradation in signal quality. Both of our schemes show a performance gain that increases with the number of served requests. This gain is caused by:

a. Both of the schemes give a more accurate placement than *Greedy-Global* since it considers the local popularity within the partition. On the other



(a) μ latency ($\mathcal{N}$=50)



(b) μ latency ($\mathcal{N}$=150)

Figure 6.2: Average latency vs. variable request rate.

hand, *Lat-CDN* assumes equal popularity of an object among all the servers and does not measure the distance between identical replicas as a demand-weighted distance. Therefore, it does not minimize the access cost.

b. Given the same popularity, our scheme favors large objects over small ones, when a DN makes the placement decision and as a result, the number of bytes traversed is reduced and the storage capacity is better utilized.

c. The low convergence time for our schemes. Since *Lat-CDN* and *Greedy-Global* require longer time to converge than *MP-DNA*, which runs in parallel by the DNs. Recall that the heuristics are evaluated in online fashion, clients' requests cannot be satisfied from the closest replica servers until the replica placement is complete. Table 6.4 shows the latency gain percentage for each heuristic over the baseline *Random* heuristic.

Table 6.4: Performance gain in latency time vs. variable number of requests for each heuristic over the baseline *Random* heuristic.

	$\mathcal{N} = 50$					$\mathcal{N} = 150$			
Requests	1000	2000	3000	4000	5000	4000	8000	12000	16000
Lat-CDN	48%	56%	50%	36%	33%	49%	52%	44%	49%
Greedy-Global	59%	55%	62%	60%	57%	53%	63%	54%	64%
SP-DNA	61%	62%	61%	65%	66%	65%	72%	67%	72%
MP-DNA	63%	64%	64%	68%	69%	68%	75%	70%	76%

To illustrate the benefit of considering the local popularity in our schemes, which is not considered in both *Greedy-Global* and *Lat-CDN*, we give an example scenario that shows the local popularity aspect. We use both Fig. 6.3 and Table 6.5. Recall that graph partitioning aims to generate equal-sized partitions such that the edge cuts between these partitions is minimized. Our schemes benefit from

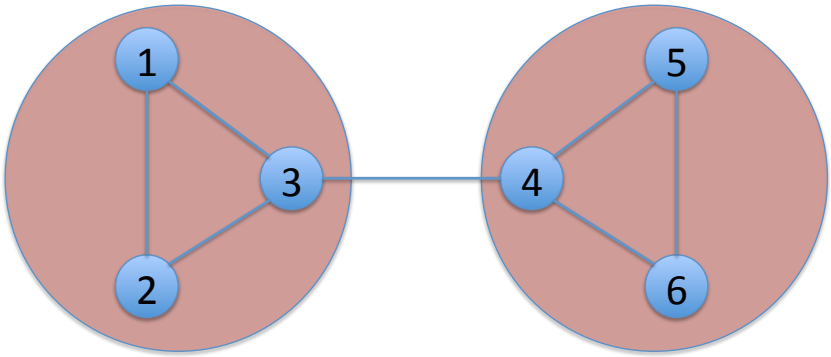


Figure 6.3: Scenario showing the benefit of considering the local popularity within a partition.

this to avoid congested links that might appear in some graphs such that the one shown in Fig. 6.3. In this scenario, suppose that the IGW to the wired Internet is at node 1, and let us assume that the distance to the origin server is undefined or for the sake of simplicity we assume that it is 50. Given the popularity and distance from each node, the *Greedy-Global* decides to place the first replica of the assumed object in node 5, since it gives the highest cost (21.2) and updates the cost row for all nodes. The second object will be placed in node 6, where the highest cost is incurred (0.29). In our schemes, we can benefit from the partitioned graph, where each replica is assigned to one partition given that it is *feasible* to place one in each. The following advantages are observed in our schemes over the *Greedy-Global*:

Table 6.5: The progress of replica placement for the *Greedy-Global* vs. our schemes.

Node ID	1	2	3	4	5	6
Popularity	9%	6%	9%	5%	40%	29%
Distance (Greedy-Global)	50	51	51	52	53	53
Cost (Greedy-Global)	4.5	3.06	4.59	2.6	**21.2**	15.37
Distance (our schemes)	1	1	0	1	0	1
Cost (our schemes)	0.09	0.06	0	0.05	0	0.29
The cost row after the first placement (violet cell) in *Greedy-Global*						
Node ID	1	2	3	4	5	6
Popularity	9%	6%	9%	5%	40%	29%
Distance (Greedy-Global)	3	3	2	1	0	1
Cost (Greedy-Global)	0.27	0.18	0.18	0.05	0	**0.29**
Distance (our schemes)	1	1	0	1	0	1
Cost (our schemes)	0.09	0.06	0	0.05	0	0.29
The cost row after the second placement (violet cells) in *Greedy-Global*						
Node ID	1	2	3	4	5	6
Popularity	9%	6%	9%	5%	40%	29%
Distance (Greedy-Global)	3	3	2	1	0	0
Cost (Greedy-Global)	0.27	0.18	0.18	0.05	0	0
Distance (our schemes)	1	1	0	1	0	1
Cost (our schemes)	0.09	0.06	0	0.05	0	0.29

6.3.3 Network Load

In this subsection, the comparison is conducted with respect to the network load, which is the total number of bytes traversing the network links (including packet headers and packet retransmissions) to serve all MCs requests during a simulation test. This is computed by multiplying the object size by the hops traversed including both success and failure requests. Two scenarios are considered to observe the factors affecting the network load. The first one considers varying the number of served requests, while the second one varies the range of object size $|\mathcal{O}_m|$.

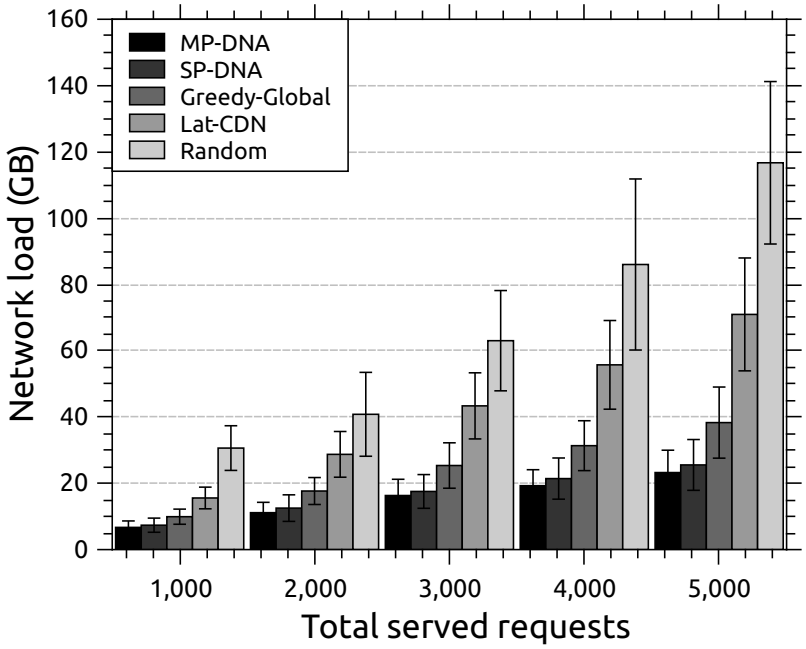
a. The overall access cost in our schemes is reduced. In our schemes, the overall access cost is 0.49, however, in *Greedy-Global* it is 0.68.

b. Since the popularity of the object in the left partition is significant, then using the *Greedy-Global* placement will direct all the requests from the left partition nodes over the single link to either node 5 or 6. This would create congestion and packet loss across this connecting link and as a result, the latency would reach its toll when such a scenario occurs in a wireless environment, where the bandwidth is limited compared to the wireline environment.

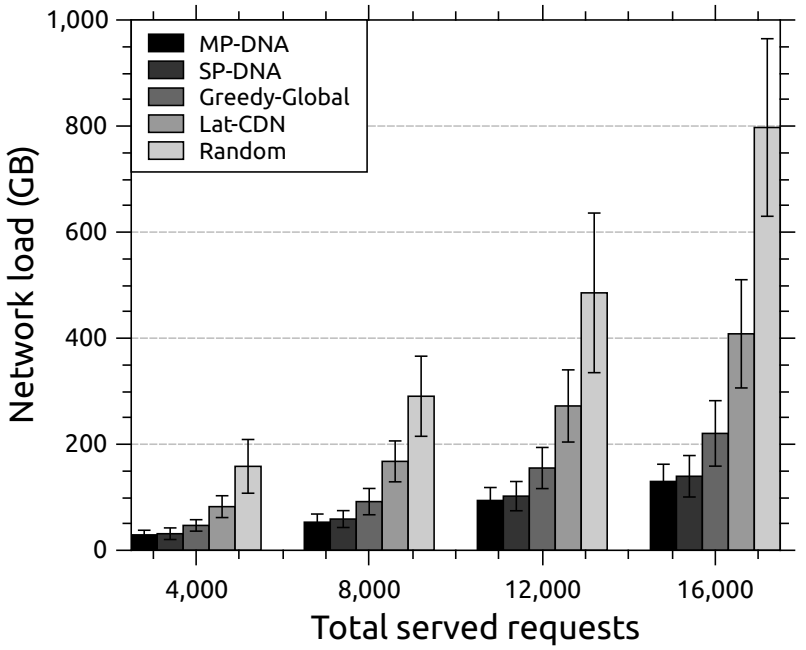
c. The placement of the two replicas in the proximity of each other will create the hotspot problem that yields to load imbalance.

Variable number of requests

Fig. 6.4 shows that the network load increases with the number of served requests for all heuristics. This is due to the incremental demand on objects during the fixed simulation time; therefore, as the number of served requests increases, the



(a)



(b)

Figure 6.4: Network load vs. different number of requests. (a) $\mathcal{N} = 50$, and (b) $\mathcal{N} = 150$.

network load increases accordingly. However, both of our schemes outperform the other heuristics significantly. Since *Greedy-Global* leads to the creation of hotspot zones, where replicas of an object are concentrated, leading to problems such as contention and congestion in such hotspot zones. As a result, packet loss increases yielding to incremental packet retransmissions. However, in our schemes, replicas are assigned to equal-sized partitions and within each partition the best placement is searched among the partition members. This helps in distributing the traffic among the partitions to avoid the aforementioned problems. *Lat-CDN* does not differentiate between objects' popularity, which yields to improper placement of replicas,

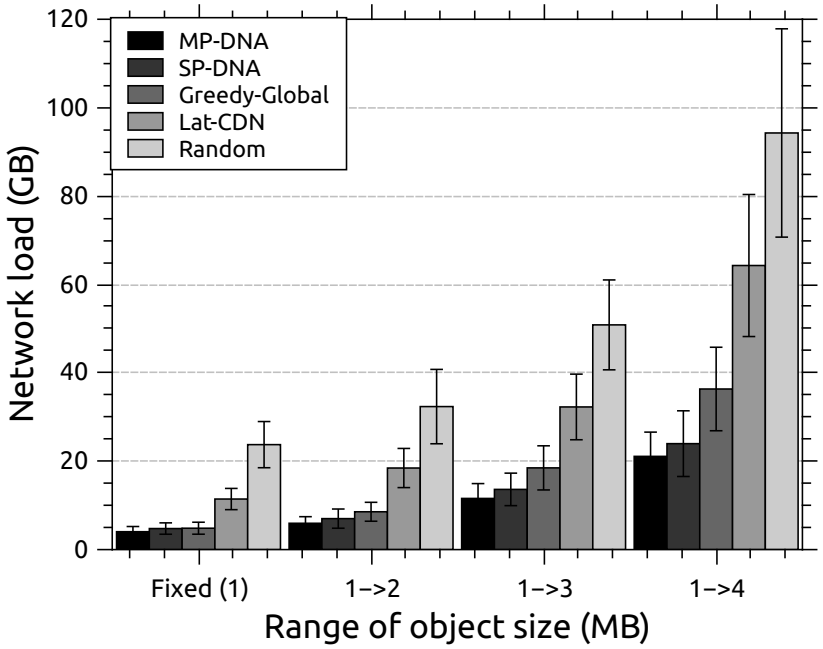
where a replica might be allocated to a server that has low popularity to that object. Table 6.6 shows the performance gain percentage for each heuristic over the baseline *Random* heuristic when varying the number of generated requests.

Table 6.6: Performance gain in network load vs. variable number of requests for each heuristic over the baseline *Random* heuristic.

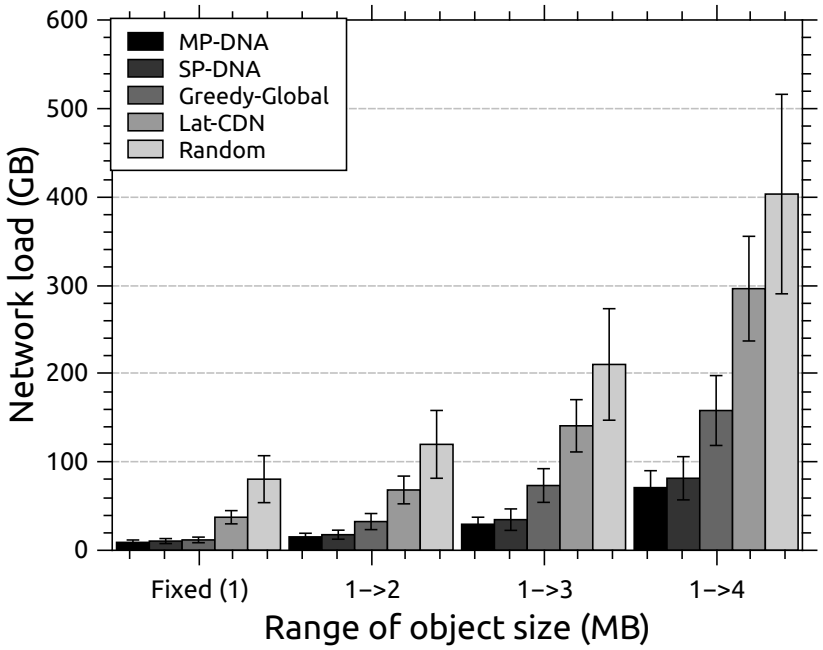
	$\mathcal{N} = 50$					$\mathcal{N} = 150$			
Requests	1000	2000	3000	4000	5000	4000	8000	12000	16000
Lat-CDN	49%	30%	31%	35%	39%	48%	42%	44%	49%
Greedy-Global	68%	57%	60%	64%	67%	70%	68%	68%	72%
SP-DNA	76%	69%	72%	75%	78%	80%	80%	79%	82%
MP-DNA	78%	73%	74%	78%	80%	82%	82%	81%	84%

Variable range of object size

In this scenario, we vary the size range of replicated objects. The ranges considered are 1 (fixed size), 1 - 2, 1 - 3 and 1 - 4 MB. The number of served requests is 5000 and 12000 for $\mathcal{N} = 50$ and 150 respectively. It can be observed from Fig. 6.5



(a)



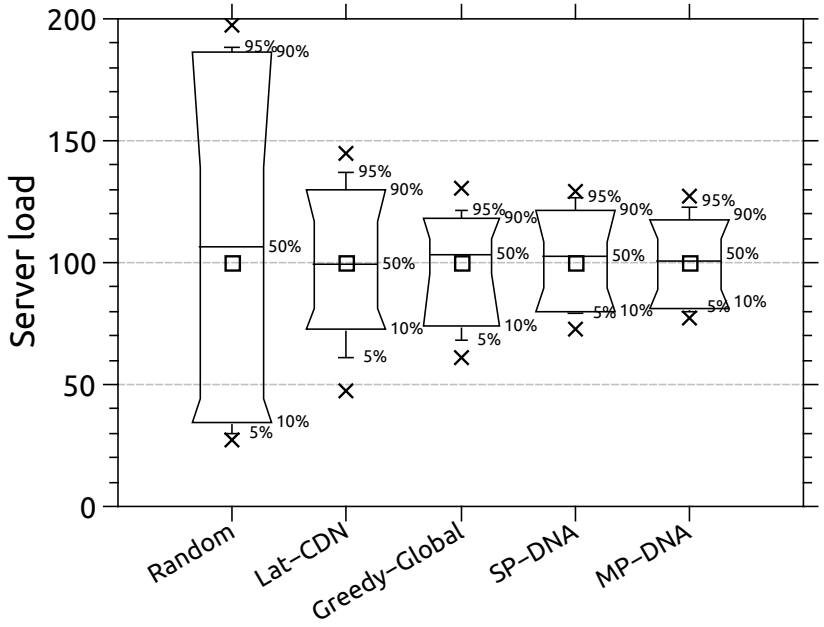
(b)

Figure 6.5: Network load for variable range of object size $|\mathcal{O}_m|$. (a) $\mathcal{N} = 50$ and 5000 served requests, and (b) $\mathcal{N} = 150$ and 12000 served requests.

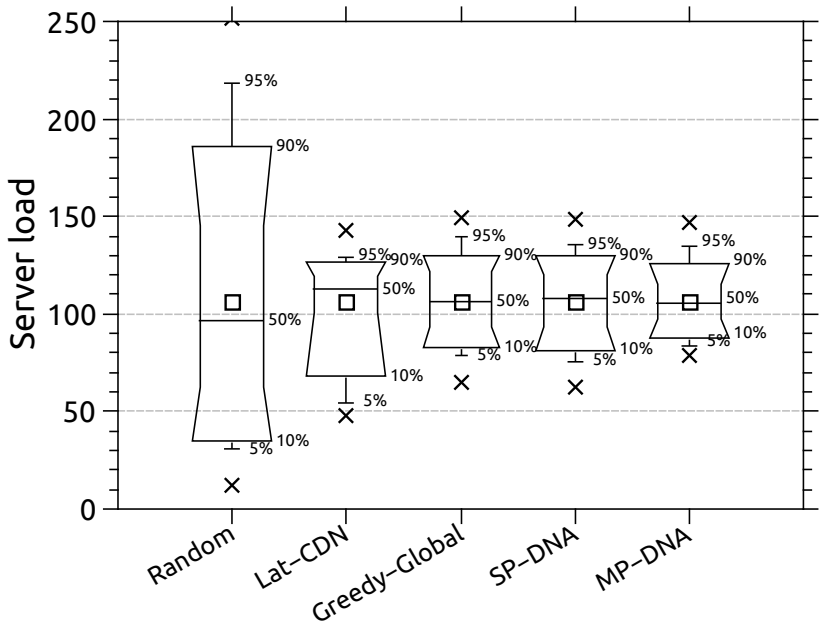
that as the range widens, the network load for all the heuristics increases. This is an inevitable result as the upper limit increases; larger objects are generated and replicated, resulting an increase in the network load. We can notice that for a fixed object size (i.e., 1 MB), *Greedy-Global* performs close to our schemes. However, as the size range increases, our schemes show a significant gain over *Greedy-Global*. This is because we consider the object size in finding the density share for each object. If we fix all the variables of Eq. (4.2) in page 76 and vary the object size, it yields that smaller objects have more replicas than larger ones. In other words, when objects have (about) similar popularity, our scheme generates more replicas for small objects in compare with large ones. Our scheme also sorts each group of object replicas in decreasing order by size. By adopting this approach, a replica server can accommodate more objects of high popularity and large size since fetching large objects is more *expensive* than the small ones. To compare with *Lat-CDN*, Table 6.7 clearly shows that our schemes have a significant performance gain over *Lat-CDN*. This gain is due to the same reason as in *Greedy-Global* (i.e., does not consider different sizes of object). Furthermore, it assumes equal popularity for an object among replica servers.

Table 6.7: Performance gain in network load vs. variable object size for each heuristic over the baseline *Random* heuristic.

	$\mathcal{N} = 50$				$\mathcal{N} = 150$			
$\lvert\mathcal{M}\rvert$ MB	1	1 - 2	1 - 3	1 - 4	1	1 - 2	1 - 3	1 - 4
Lat-CDN	52%	43%	37%	32%	54%	43%	33%	27%
Greedy-Global	80%	74%	64%	61%	85%	73%	65%	61%
SP-DNA	80%	78%	73%	75%	87%	85%	84%	80%
MP-DNA	83%	82%	77%	78%	89%	87%	86%	82%



(a)



(b)

Figure 6.6: Distribution of server load for variable number of requests. (a) $\mathcal{N} = 50$ and 5000 served requests, and (b) $\mathcal{N} = 150$ and 16000 served requests.

6.3.4 Server Load

In this subsection, we compare our schemes with the other heuristics with respect to the load on replica servers. This represents a server's share among all the requests during a simulation test. This scenario is similar to the one in subsection 6.3.3. The difference is that the number of served requests is fixed for both network sizes to observe the load distribution on the servers. Each simulation test is run for a specific number of requests for all the heuristics and for both network sizes. The results in Fig. 6.6 represent the load distribution on the servers for the largest number of requests (i.e., 5000 and 16000). We chose to show these two simulation tests among the rest because they give more accurate results. The small square in each box plot represents the ideal situation of equal load distribution. We can notice that the worst server load distribution is the one for the baseline *Random* heuristic. This is due to its random behavior. textitLat-CDN has a much better load distribution than *Random* because servers cooperate to place objects with the high latency. However, it does not perform better than *Greedy-Global* and both of our schemes because it leads to improper replica placement. In compare with *Greedy-Global*, we can notice that our schemes have a better load distribution than *Greedy-Global*. This comes as a result of hotspot creation that we explained previously. However, the traffic in *Greedy-Global* will be concentrated in that partition; therefore, two problems will arise. The first problem is the huge traffic within that partition yielding to problems such as the contention to access the wireless medium, intra-flow and inter-flow interference, packet loss and congestion, which will degrade the link quality. The second problem is the load imbalance between replica servers. However, our schemes can achieve a better load balance than *Greedy-Global*, since requests can be forwarded to replica servers in the neighboring partitions. For *Greedy-Global*, this might not be a problem as long as there is enough bandwidth and

high capacity CDN servers, but in the wireless environment these resources are scarce.

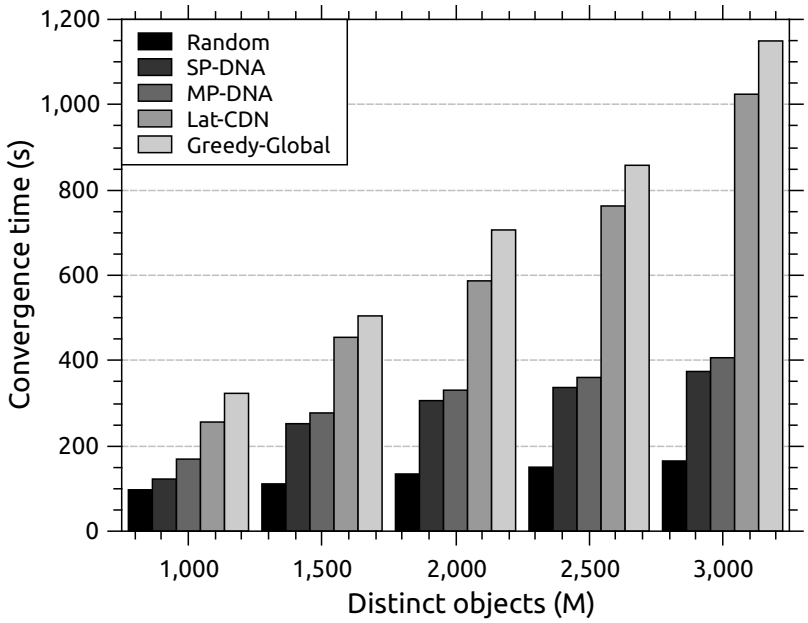


Figure 6.7: Convergence time for variable number of distinct objects $\mathcal{M}$ and $\mathcal{N} = 150$.

6.3.5 Convergence Time

In this subsection, we compare our schemes with the other heuristics with respect to convergence time. The network size considered is $\mathcal{N} = 150$ and the number of distinct objects ($\mathcal{M}$) varies from 500 to 3000. The Storage Capacity ($\mathcal{SC}$) is set to 1024 MB. Fig. 6.7 clearly shows that the *Random* algorithm has the best convergence time. This is due to the local decision on the replica placement that does not have any form of cooperation. On the other extreme, *Greedy-Global* and *Lat-CDN* demonstrate long time requirements to converge. This is because both of them are centralized and require heavy sorting operations. Both of them

require recalculating the cost function for every *Object-Server* assignment after every replica placement. We can notice the correlation between $\mathcal{M}$ and $\mathcal{N}$ as they increase, the growth of the convergence time for both increases exponentially. This indicates that both do not scale well with respect to $\mathcal{M}$ and $\mathcal{N}$. The performance gain for *Lat-CDN* over *Greedy-Global* is because the latter requires collecting popularity statistics about objects, which is not required by the former algorithm. In contrast, our schemes are distributed and therefore, the burden of the placement decision is performed in parallel by the DNs showing a logarithmic increase in convergence time with respect to $\mathcal{M}$ and/or $\mathcal{N}$. This is because for a specific $p_m(\tau + 1)$, there will be an equal number of DNs, where each DN locates a single replica of $\mathcal{O}_m$ within its partition. As the number of replicas increases, the partition size ($\mathcal{N}'$) decreases yielding a decrement in convergence time. Overall, we observed that the modified versions of *SP-DNA* and *MP-DNA* require longer time to converge than the basic versions in the previous two chapters. This is due to the consideration of local popularity that forwards the placement job to other larger partitions yielding to extra time requirements.

6.3.6 Communication Overhead

Now we observe the communication overhead for each scheme. Three network sizes ($\mathcal{N} = 50$, 150 and 300) were considered to quantify the scalability of each scheme. In the simulation runs, we considered placing the replica placement entity for both *Greedy-Global* and *Lat-CDN* in a centroid node to reduce the wireless transmissions. The results in Fig. 6.8 reveal that *Greedy-Global* incurs the highest overhead. This is caused by the hop-by-hop forwarding of popularity messages to the central entity, which finds the set of object replicas for each node and then forwards to each node its own replica set. *Lat-CDN* works similar to *Greedy-Global* except that it does not require the collection of popularity statistics,

therefore, the overhead is reduced by approximately half. The agglomeration approach used by both of our schemes significantly reduces the message overhead since the partition members are in the proximity of their DN, the wireless transmissions traverse a short distance. As we go up the hierarchy/Map-List, the agglomerated message might travel a longer distance, however, it combines multiple popularity messages from its descendants/subset partitions. On the other direction, when the CM assigns the placement job, it *batches* multiple object IDs in a multicast message that is forwarded to the corresponding DNs. We observed that the modified versions of *SP-DNA* and *MP-DNA* require incur more message overhead than the basic versions in the previous two chapters. This is due to the consideration of local popularity that forwards the placement job to other larger partitions yielding to extra message transmissions.

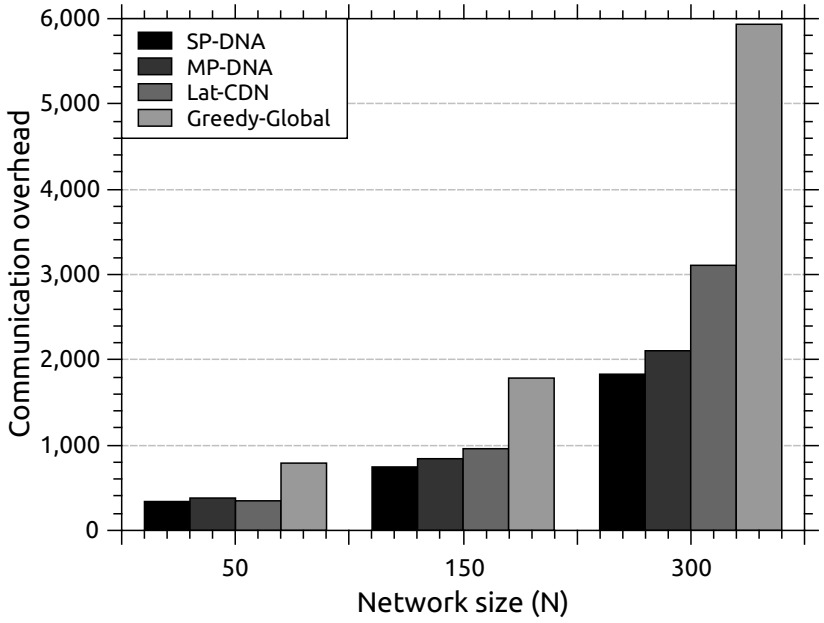


Figure 6.8: Comparison between different schemes in terms of communication overhead.

6.4 Summary

In this chapter, we turned our attention to scenarios where we might have diversity in content popularity. This means that the popularity of objects is not uniformly distributed among mesh routers. The basic schemes in both chapters 4 and 5 were not designed to consider such scenarios. Henceforth, we modified both of the schemes to enhance the placement by introducing local popularity within the partition. A delegate node decides to place a replica in its partition when it finds that it is feasible. If not, it forwards the placement job to a larger partition. As a result, the schemes are capable of making accurate placement by comparing the local popularity with a feasibility index that can be tuned with a threshold coefficient. We have shown how our schemes exploit graph partitioning to avoid congested links between partitions and avoid the creation of hotspot zones. The schemes take into account the factors of object popularity and size to compute the number of replicas per object. The obtained simulation results show that the proposed replication schemes can significantly improve network performance in terms of latency, hop-count distance, network load. However, the improvement is slight –compared to *Greedy-Global*– in the server load. We also note that there is a slight increase in the convergence time and communication overhead –compared with the schemes in chapters 4 and 5– due to the forwarding of infeasible objects to larger partitions.

Chapter 7

Link-Quality Based Placement

The challenge of minimizing object access cost is more serious in WMNs than in wired networks. Replica placement algorithms that were designed for CDN servers consider minimizing the number of hops between the requesting client and the nearest replica server. However, the hop-count metric exhibits poor performance in WMNs since it does not take the time varying link-quality into account [102]. Minimizing the hop-count might lead to the selection of low SNR links or links with reduced transmission rates, resulting in low throughput. A replica placement decision should consider the challenges surrounding the link-quality such as the contention between neighboring mesh nodes for the wireless channel and the interference from the adjacent wireless links that lead to a significant reduction in throughput over a long path. Therefore, we consider link-quality metrics to improve the selection of high quality links between the requesting node and the replica server in both the replica placement decision and the replica server selection.

In the previous chapters, we proposed two distributed content replication schemes. However, the schemes aim to minimize the simple hop-count metric, which does not reflect the underlying MAC/PHY link capacity. Other link-

quality routing metrics prove to be more efficient than the simple hop-count such as Expected Transmission Count (ETX) [24], Expected Transmission Time (ETT) [26] and Minimum Loss (ML) [25]. This enables us to accurately gauge the client's object access cost. The central idea behind this enhancement is to use the long-term link-quality values for the replica placement decision, which is performed periodically. Furthermore, for replica server selection, we use the instantaneous link-quality values to satisfy a client's request.

The problem we aim to address in this chapter can be described as follows. We consider an infrastructure WMN, where nodes (MRs) are deployed in static locations. In such networks, nodes do not join/leave the network dynamically. However, the wireless channel state between two neighbors may oscillate frequently due to various reasons. Since object replica placement is a costly operation, a replica must remain for a reasonable period of time. Most of the works in the wired/wireless literature consider minimizing the number of hops or the Euclidean distance in the placement decision. However, we argue that this approach might not yield to efficient placement since a shortest distance can lead to lossy/congested links. We motivate our work by showing the performance gain when the replica placement considers the long-term link cost that represents its throughput or channel capacity. The short-term link cost may fluctuate frequently, whereas its long-term cost varies much slowly. In WMNs, the contention between neighboring mesh nodes for the wireless channel, together with the interference from the adjacent wireless links, results in a significant reduction in throughput over a long lossy path. Therefore, MRs that are far from the IGWs suffer from long access latency and low throughput. We consider minimizing the demand-weighted distance between the requesting node and the replica server. Another aspect that we consider in this chapter is the estimation/prediction of object popularity for the next replication period. In the previous chapters, we

based our estimation only on the previous period. However, content popularity (usually) increases by time and reaches a peak and then starts to drop down.

Our contribution in this chapter can be summarized as follows:

a. Using advanced link-quality metrics to accurately gauge the client's object access cost using the long-term link-quality values for the replica placement decision, which is performed periodically. Furthermore, for replica server selection, we use the instantaneous link-quality values to satisfy a client's request. We turn our attention to exploiting link-quality information to augment both replica placement and server selection decisions. We investigate common WMN link-quality routing metrics such as ETX, ETT and ML.

b. We improve the popularity estimation model using the D-EWMA smoothing to predict the number of replicas for the next replication period.

c. Modify and implement relevant schemes that were designed to consider the hop-count metric as an objective to be minimized. Therefore, we modify these schemes to consider the previously mentioned link-quality metrics. By conducting extensive simulation tests, we compare the performance of our scheme with the most relevant ones and show that our schemes perform better than the other schemes and significantly improve the system performance.

7.1 Link-Quality Routing Metrics

In this section, we discuss the used routing metrics in the improved versions of our schemes. It is very important in our scenario to select the suitable routing

metric since it is used in the cost function that we want to optimize in the replica placement heuristic on one side, and replica server selection on the other side:

a. **Expected Transmission Count (ETX)**: This is the expected number of transmissions a node requires to successfully transmit a packet to a neighbor. To find ETX [24], each node periodically broadcasts packet probes that contain the number of received probes from each neighbor. A node finds the ETX of a link with its neighbor using the delivery ratio of probes sent on the forward (*df*) and reverse (*dr*) directions:

$$ETX = \frac{1}{d_f \times d_r} \tag{7.1}$$

The chosen route is the one with the lowest sum of ETX along the route to the destination. However, probing packets are smaller than data packets; hence, if the network is operating at high rates, the performance of ETX becomes low because it does not distinguish links with different bandwidths and does not consider data-packet sizes.

b. **Minimum Loss (ML)**: This metric [25] is based on probing packets to compute the delivery ratio. It finds the route with the lowest end-to-end loss probability. Instead of adding (used in ETX), it multiplies the delivery ratios of the links in the reverse and forward directions to find the best path. The best path is the one with the maximum end-to-end success probability.

c. **Expected Transmission Time (ETT)**: To overcome the ETX drawbacks, ETT was proposed [26] to improve the performance of ETX in multi-radio WMNs that support different data rates by including the link bandwidth in its computation:

$$ETT = ETX \times \frac{S}{B} \tag{7.2}$$

Where S is a fixed data-packet size (1137 Bytes) and B is the link bandwidth. The bandwidth is estimated periodically by transmitting a sequence of two packets unicasted, a small one followed by a large one (137 and 1137 bytes). Each neighbor measures the inter-arrival period between the two packets and reports it back to the sender. The bandwidth is the size of the large packet divided by the minimum delay received for that link.

7.2 Modified *SP-DNA* and *MP-DNA* Schemes

In this section, we describe the modifications made for both the *SP-DNA* and *MP-DNA* schemes to consider the placement based on the long-term link cost. These modifications are as follows:

- In the substep **During the τ period** of the **Content Replication and Placement Phase** (see page 75 for *SP-DNA* and page 102 for *MP-DNA*). In addition to collecting statistical information about different objects request frequencies. Each node will also observe the link-quality (e.g., ETX, ETT and ML) values with its neighboring nodes by logging the values of the used metric for the links between the partition nodes.

- In the substep **At the end of the τ period** (pages 75 and 102) of the **Content Replication and Placement Phase**. In this substep, every node finds the *trimmed mean* value of the link-quality metric with the neighboring nodes by excluding *outlier* values. Since the replica placement is intended for long periods, using the current value of the link-quality metric will not reflect the actual distance cost. Therefore, we believe that using the trimmed mean can give a more precise estimation of fluctuating wireless links that captures the long-term link cost instead of using the mean or instantaneous link cost. Anomalous outlier values can completely change the mean, thus

putting the link cost at big risk of errors. Therefore, it is important to identify and eliminate them as follows:

We let x_1, x_2, ..., x_s be a sample of size s on measurement of a particular link-quality metric. Then the $100\alpha\%$ trimmed mean is defined as:

$$\overline{X}_t = \frac{\sum_{i=t+1}^{s-t} x_i}{(s(1-2\alpha))} \tag{7.3}$$

where $\alpha \in (0,1)$; $t = [s\alpha + .4]$. To simplify, we take t to be the floor of $(s\alpha + .4)$ as an approximation [103]. Every node computes $\overline{X}_t$ with its neighboring nodes to represent the range of sample points unaffected by outliers. We simply ignore t of the lowest and t of the highest sample points. Then the following sub-steps are performed:

(a) The partition members of the lowest level, fine-grained DNs forward their object frequency list and the computed link-quality trimmed mean $\overline{X}_t$ with the member's neighbors to their DN(s).

(b) For *SP-DNA*, each DN aggregates the received lists from its children along with its own list, stores the resulting list and then forwards it to its parent DN in the DBT. For *MP-DNA*, each DN aggregates the received lists from the partition members, removes any redundant lists, stored the resulting list and then forwards it to the CM.

(c) The CM creates a *Replica-List* (RL) in the form of a 4-tuple that contains the entries $\langle p_m(\tau+1), \mathcal{O}_m, |\mathcal{O}_m|, \lambda_m(\tau)\rangle$ grouped by $p_m(\tau+1)$ in a decreasing order to prioritize the most popular objects. Within each group, the objects are sorted by the objects size $|\mathcal{O}_m|$ in a decreasing order.

- In the **Replica assignment and placement** (see page 121 since we use the modified schemes in the previous chapter that consider the local popularity). We modify both of Alg. 7 (page 122) and Alg. 8 (page 123) by replacing

the distance $d_{i'j'}$ with $\Gamma(\overline{X}_t)_{i'}^{j'}$. This distance ($\Gamma(\overline{X}_t)_{i'}^{j'}$) represents the sum (except for ML, which is multiplicative) of the long-term link-quality between i' and j'. The DN computes the total cost for every partition node and assigns $\mathcal{O}_m$ to the node that minimizes the total cost.

7.3 Popularity Estimation

In our scheme, the size of the data set, its periodic evaluation, and other parameters depend on the application, workload and the request pattern. However, we can evidence that existing approaches which take into account just the basic statistics on past resource accesses [104, 105] can achieve good results in the context of traditional Web-based applications, where the resource popularity changes slowly and according to known patterns, which might not achieve acceptable results when applied in WMNs applications. The main motivation of this section is that estimating the content popularity is a dimension to be considered in our proposed schemes to achieve more accurate prediction of the content popularity for the next period using the statistics from previous periods and the trend in objects' popularity.

There is a plethora of estimation methods that can be used in predicting the number of replicas for the next replication period. A class of these methods is the *Exponential Smoothing* method. Exponential smoothing is a method that can be applied to time series data to produce smoothed data for presentation, or to forecast future data. The time series data are a sequence of observations. The phenomenon under observation can be a random process, or it can be a noisy and/or orderly process. Some of the methods under this classification are:

a. Simple moving average: The past observations are weighted equally for the previous n datum points.

b. Cumulative moving average: This considers all the observed data points until the current one.

c. Weighted moving average: This average has multiplying factors to give different weights to data at different positions in the sample window with the highest weight for the most recent datum points.

d. Exponential smoothing [106] or exponentially weighted moving average (EWMA) can be formulated as follows:

$$s_1 = x_0 \tag{7.4}$$

$$s_\tau = \beta x_{\tau-1} + (1-\beta)s_{\tau-1}, \tau > 1 \tag{7.5}$$

where β is the *smoothing factor* such that $0 < \beta < 1$. This means that s_τ is a simple weighted average of the previous observation $x_{\tau-1}$ and the previous smoothed value $s_{\tau-1}$. Larger values of β reduce the level of smoothing, and in the case where $\beta = 1$, the output series is just the same as the original series. We can note that this method is easily applied and provides a smoothed statistic as soon as two observations are available.

e. Double Exponential Smoothing (D-EWMA) [107]: EWMA smoothing does not work well when there is a trend in the observed data points. In such case, the D-EWMA takes into account the trend in the data sequence. Double exponential smoothing works as follows:

The data sequence of observations is represented by $\{x_\tau\}$, beginning at time $\tau = 0$. We use $\{s_\tau\}$ to represent the smoothed value for time τ, and $\{b_\tau\}$ is our best estimate of the trend at time τ. The output of the D-EWMA algorithm is written as F_{t+m}, an estimate of the value of x at time $t+m$, $m > 0$ based on the raw data up to time τ. D-EWMA is given by the

formulas:

$$s_1 = x_1 \tag{7.6}$$

$$b_1 = x_1 - x_0 \tag{7.7}$$

Then for $\tau > 1$

$$s_\tau = \beta x_\tau + (1 - \beta)(s_{\tau-1} + b_{\tau-1}) \tag{7.8}$$

$$b_\tau = \gamma(s_\tau - s_{\tau-1} + (1 - \gamma)b_{\tau-1} \tag{7.9}$$

where β is the data smoothing factor, $0 < \beta < 1$, and γ is the trend smoothing factor, $0 < \gamma < 1$. To forecast beyond x_τ

$$F_{\tau+m} = s_\tau + mb_\tau \tag{7.10}$$

Setting the initial value b_0 is a matter of preference. An option that can be used is $(x_n - x_0)/n$ for some $n > 1$. Note that F_0 is undefined as there is no estimation for time 0, and according to the definition $F_1 = s_0 + b_0$, which is well defined, thus further values can be evaluated.

To provide a more accurate and robust prediction of objects popularity for the next replication period, we adopt the use of the D-EWMA method since it considers the trend in the data sequence, and typically, object popularity increases by time until it reaches a peak and then starts to drop. In our previous experiments, the estimation was only considering the last replication period. Therefore, we anticipate that the prediction of future popularity can reflect on the computed number of replicas by the CM.

7.4 Simulation Experiments

In the next section, we examine the feasibility of our scheme by comparing it to relevant placement heuristics through detailed simulations. The methodology

for the conducted simulation experiments is similar to the one in page 80. We implemented the modifications on our schemes using OMNeT++ [94] simulator with Inetmanet [95]. Inetmanet provides an implementation for different MANET routing protocols with the support for multiple link-quality metrics including hop-count, ETX, ETT and ML. We simulated a stationary WMN for different scenarios in different mesh topologies. We used the OLSR [100] routing protocol, which is implemented by Inetmanet augmented with the link-quality metrics (i.e., hop, ETX, ETT and ML). Unless otherwise stated, the default values of our simulation are presented in Table 7.1. Three network sizes are considered

Table 7.1: Default simulation parameters

Parameter	Default Value
Number of MRs ($\mathcal{N}$)	50, 150 and 300
Radio interface	IEEE 802.11g
Link rate	54 Mbps
Transmission range	140m
Zipf-like parameter α	0.80
Threshold coefficient θ	1
Carrier frequency	2.4 GHz
Max. Tx power	20 dBm
Noise level	-110 dBm
Channel model	Rayleigh
Percent of trimmed values t	10%
$\mathcal{M}$	10000
$\lvert\mathcal{O}_m\rvert$ range	$3 \Rightarrow 10$ MB
τ	240 minutes
$\mathcal{SC}$	2 GB

$\mathcal{N} = 50$, 150 and 300 deployed uniformly at random in areas of 700 m $\times$ 700 m, 1500 m $\times$ 1500 m and 2250 m $\times$ 2250 m respectively. Each node is having four active mesh clients. Requests follow the Zipf-like distribution [7, 108] and the value of the Zipfian parameter α was chosen to be 0.83 based on measurements on web traces [108]. Object request model is similar to the one in [64], where each node generates a single stream of read-only queries. After a query is sent out, the node does not generate new query until the query is served. A MC requests an object in a period ranging between 5 and 15 minutes on average. The upper bound is caused by the *Random* scheme. The replication period τ = 240 minutes (simulation time) of mesh client activity. $\mathcal{M}$ was set to 10000 with an object size ranging between 3 and 10 MB [101] following a uniform distribution from which each client makes a request based on the Zipf-like distribution. The mesh router storage size SC is set to 2 GB and the application traffic is created using FTP traffic generator. The average distance between a MR and its neighbor is 100 meters. MRs are equipped with IEEE 802.11g radio functioning at 54 Mbps link rate and a transmission range of 140 meters. The simulation tests are carried out in a transient state system and online fashion such that object replica placement is performed whilst clients' requests are generated. To serve a request, four link-quality metrics are considered, namely hop-count, ETX, ETT and ML. Requests are served from the closest replica server depending on the instantaneous value of the link-quality metric used. Although the heuristics we are comparing with use the hop-count metric, we find it more equitable to use the other metrics in our comparison. Therefore, the distance considered in the placement and server selection decisions for the other schemes reflects the used link-quality metric. For this purpose, we modified the schemes we are comparing with to consider the long-term and instantaneous link metric for both the placement and retrieval decisions.

7.5 Results and Discussions

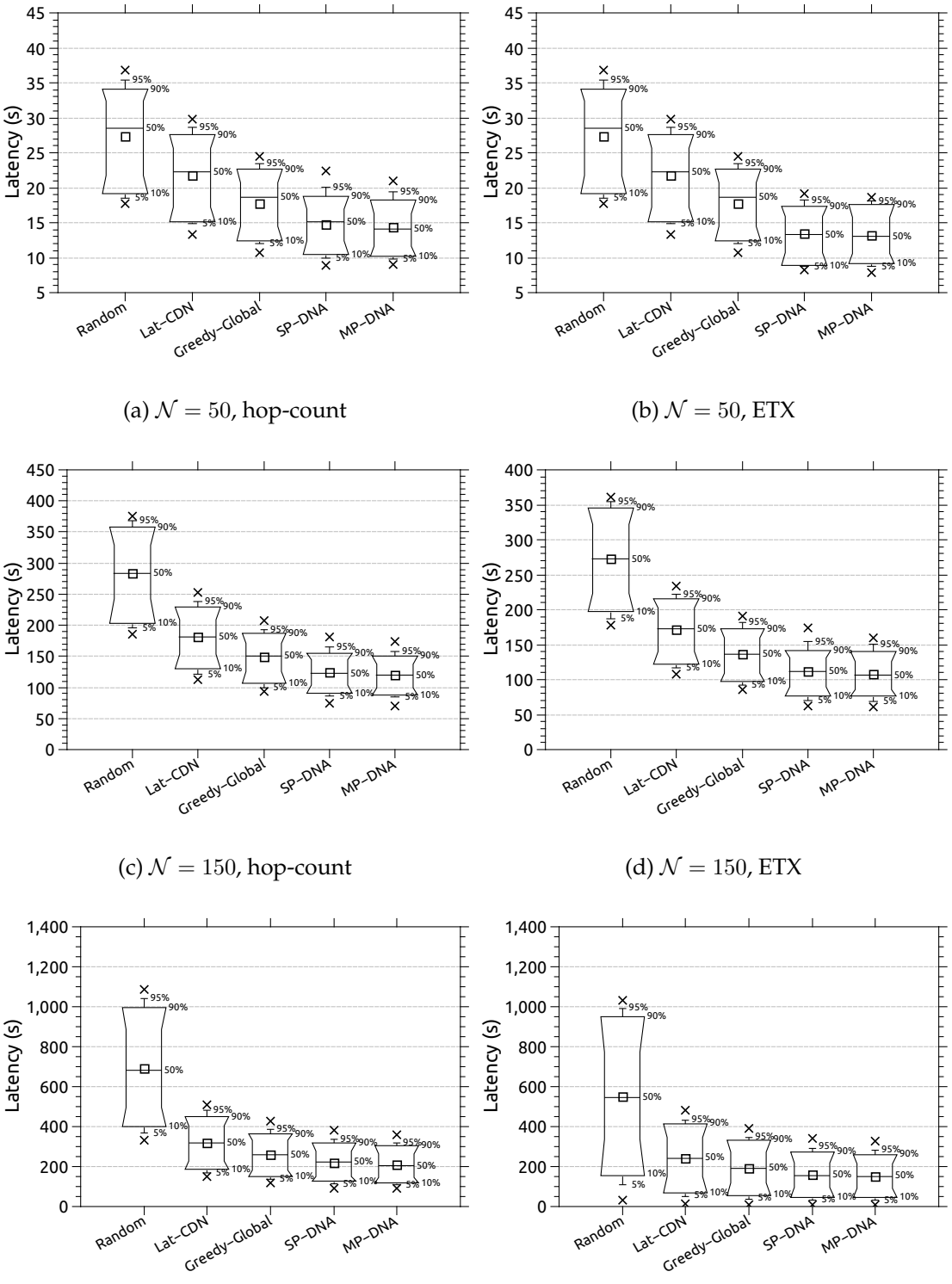
In this section, we present the simulation results and discuss them according to the performance metrics described in each sub-section.

7.5.1 Average Latency Time

In this subsection, we compare the performance of our schemes and the other heuristics with respect to the latency time observed by mesh clients. We ran extensive simulation tests for each scenario with different values of $\mathcal{N}$ and different topologies. The results in Fig. 7.1 depict a comparison for six different scenarios using the hop-count and ETX metrics, while the results in Fig. 7.2 show a comparison for another six scenarios using the ML and ETT metrics.

It can be noticed that as the network size increases, the delay increases proportionally. This can be clearly shown when $\mathcal{N} = 300$ for all the heuristics since the request rate increases dramatically adding extra load on the network. Another reason is that the distance between the requesting client and the replica server increases. This aggravates the channel competition between the upstream and downstream traffic yielding packet delay, loss, and retransmission due to channel congestion and degradation in signal quality. It can be noticed that the worst performance is obtained using the simple hop-count, which does not reflect the channel quality leading to lossy/congested links and as a result, yields to packet delay, loss, and retransmission due to channel congestion and degradation in signal quality.

The results in both Fig. 7.1 and Fig. 7.2 clearly show that all the heuristics perform best using the ETT metric, which employs the link-rate information to represent the wireless link-quality more precisely than ETX and ML; and definitely much more precise than the simple hop-count. We can note that



(a) $\mathcal{N} = 50$, hop-count

(b) $\mathcal{N} = 50$, ETX

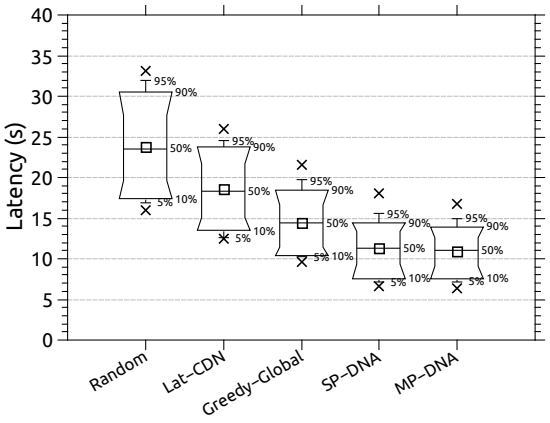
(c) $\mathcal{N} = 150$, hop-count

(d) $\mathcal{N} = 150$, ETX

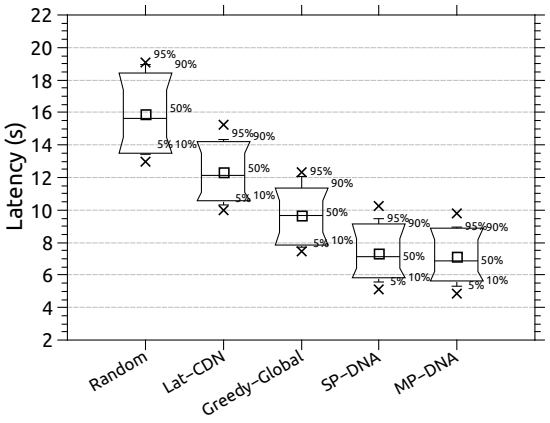
(e) $\mathcal{N} = 300$, hop-count

(f) $\mathcal{N} = 300$, ETX

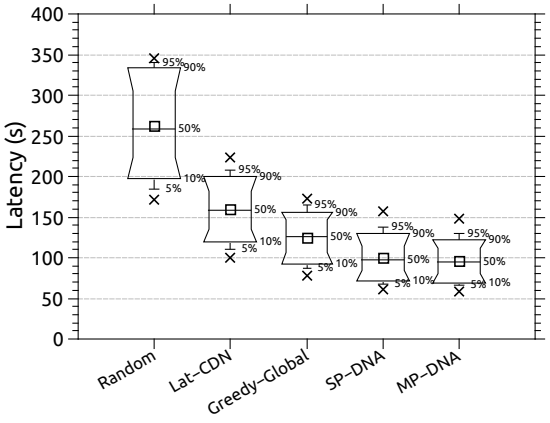
Figure 7.1: Latency time statistics for different network sizes $\mathcal{N} = 50$, 150 and 300 using the hop-count and ETX link-quality metrics. The figure shows the Min, Max, Mean and different percentiles.



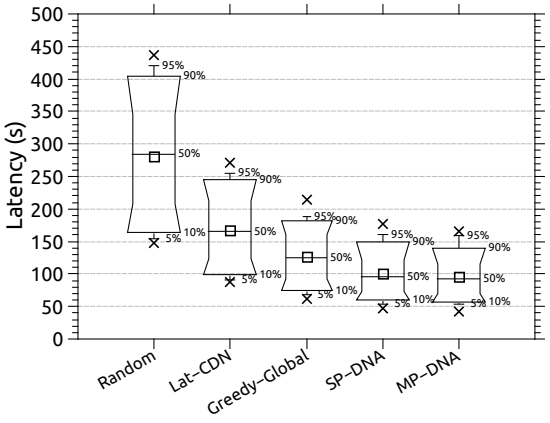
(a) $\mathcal{N} = 50$, ML



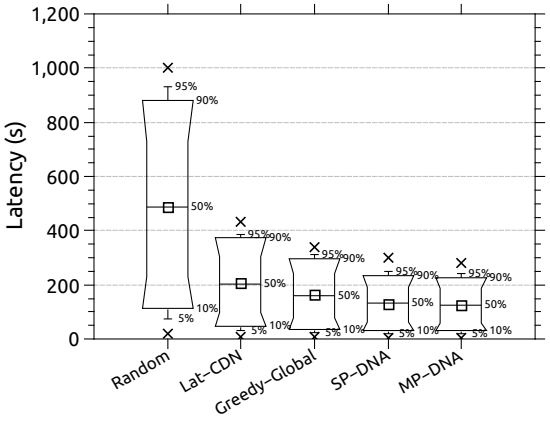
(b) $\mathcal{N} = 50$, ETT



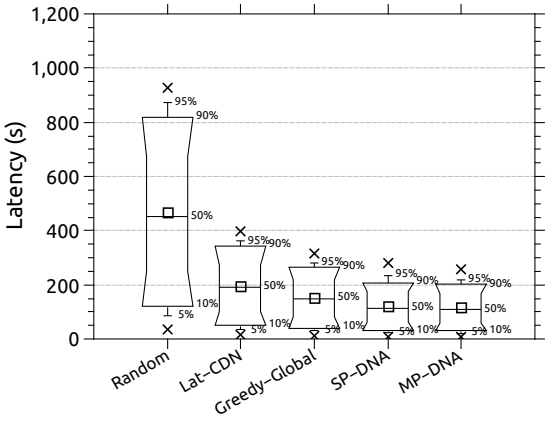
(c) $\mathcal{N} = 150$, ML



(d) $\mathcal{N} = 150$, ETT



(e) $\mathcal{N} = 300$, ML



(f) $\mathcal{N} = 300$, ETT

Figure 7.2: Latency time statistics for different network sizes $\mathcal{N} = 50$, 150 and 300 using the ML and ETT link-quality metrics. The figure shows the Min, Max, Mean and different percentiles.

our schemes outperform the other heuristics. The performance gain is caused by two factors; the first is that increasing the number of requests implies that clients are accessing a wider variety of different objects with different sizes. The performance gain is caused by:

a. Given the same popularity, our scheme favors large objects over small ones when a DN makes the placement decision and as a result, the number of bytes traversing is reduced.

b. Our schemes give a more accurate placement than *Greedy-Global* since it considers the local popularity within the partition.

c. The fairness towards small objects in terms of the number of replicas per object due to the division by the object size.

d. In compare with *Lat-CDN* that assumes similar popularity of an object among all the servers, the resulting placement is not efficient that places unpopular or less popular objects in locations that do not exhibit demand for such objects.

e. The shorter convergence time for our schemes. Since *Lat-CDN* and *Greedy-Global* require longer time to converge than our schemes. This might redirect clients' requests to servers far from the MR that the client is associated with during the placement time.

7.5.2 Total Number of Packet Collisions

In this subsection, the comparison is conducted with respect to the number of packet collisions. This refers to the total number of colliding packets resulting either from simultaneously receiving multiple packets from the contention to

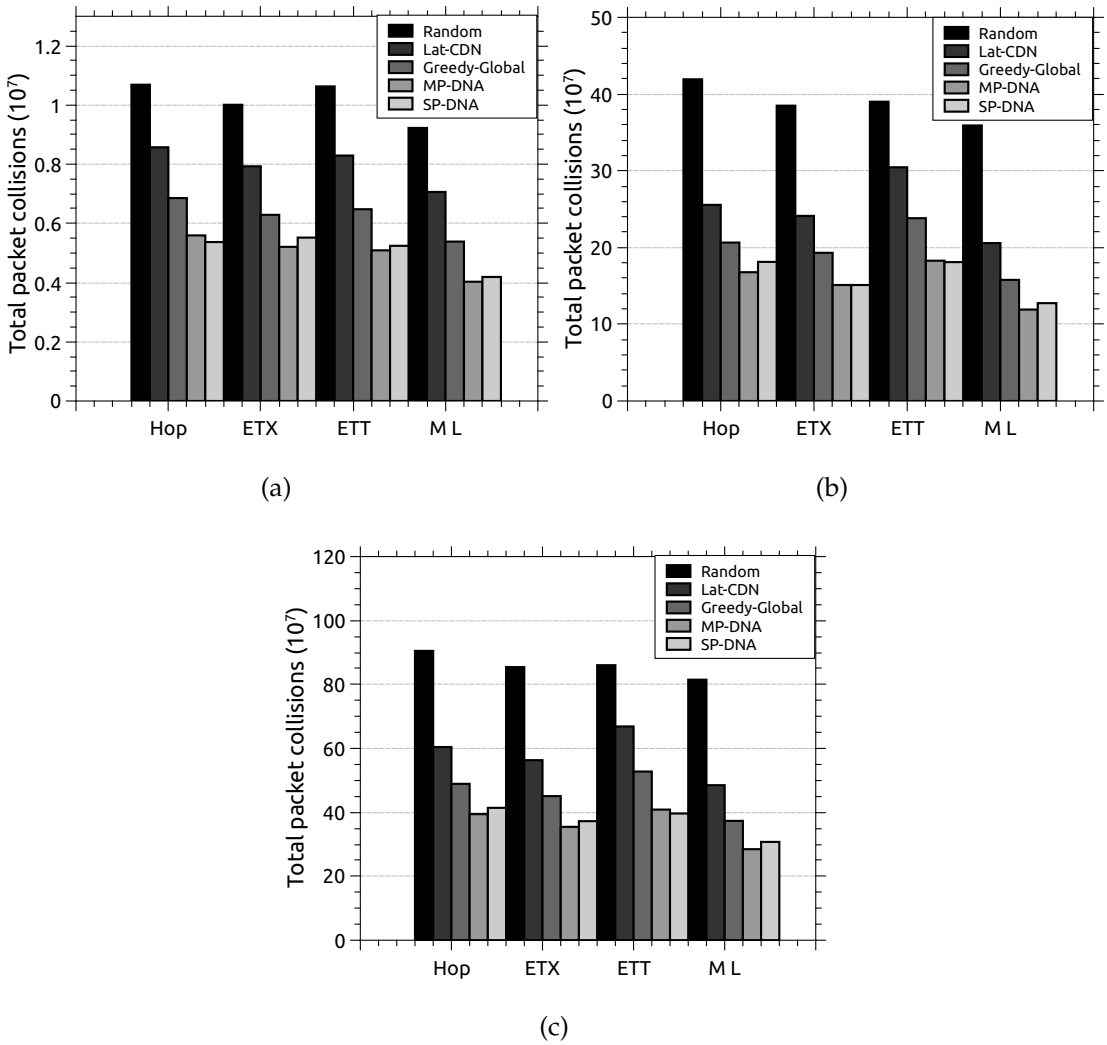


Figure 7.3: Total number of packet collisions (x10^7) for (a) $\mathcal{N} = 50$, (b) $\mathcal{N} = 150$ and (c) $\mathcal{N} = 300$.

access the wireless medium during a simulation run. Since MRs are typically stationary, link failure due to mobility is rare. However, transmission fails due to packet collisions, interference, or inadequate link rate selection. We can notice from Fig. 7.3 that the packet collision increases drastically for the baseline *Random* heuristic as a result of the long distance traveled by the packets, which degrades the network capacity. It can be noticed that Our schemes can significantly reduce the number of packet collisions compared to the other heuristics. This performance gain is because *Greedy-Global* might lead to the creation of hotspot zones, where replicas of an object are concentrated yielding to problems such as contention and congestion in these hotspot zones that increases packet collision. The increased packet loss in *Lat-CDN* comes as a result from the assumption of similar object popularity among replica servers. This assumption is not practical in WMNs that do not have sufficient bandwidth as in CDN servers yielding improper and inefficient placement of unpopular objects in nodes that can benefit from hosting popular objects with respect to their MCs demands. As a result, the distance traveled is increased and a node's $\mathcal{SC}$ is inefficiently utilized. However, in our scheme, the traffic is distributed among the partitions to avoid the aforementioned problems. We note (See Table 7.2) that in terms of packet collision, ML performs best since it is based on using the path that introduces the minimum loss rate by multiplying the success probability of each link along the path. This avoids lossy links and therefore, it reduces packet collisions. The results also reveal that the ETT metric incurs more packet collisions. This is because ETT injects extra probes of data-packets, packet collision probability increases.

The increased packet loss in *Lat-CDN* comes as a result from the assumption of similar object popularity among replica servers. This assumption is not practical in WMNs that do not have sufficient bandwidth as in CDN servers

Table 7.2: Total number of link layer packet collisions (x10^7).

	$\mathcal{N}$ = 50				$\mathcal{N}$ = 150				$\mathcal{N}$ = 300			
Metric	Hop	ETX	ETT	ML	Hop	ETX	ETT	ML	Hop	ETX	ETT	ML
Random	1.07	1.00	1.06	0.92	41.92	38.49	39.00	35.91	90.50	85.43	86.02	81.49
Lat-CDN	0.86	0.79	0.83	0.71	25.56	24.12	30.46	20.58	60.42	56.32	66.86	48.49
Greedy-Global	0.69	0.63	0.65	0.54	20.65	19.31	23.82	15.78	48.91	45.08	52.76	37.33
MP-DNA	0.56	0.52	0.51	0.40	16.78	15.12	18.28	11.92	39.44	35.46	40.86	28.47
SP-DNA	0.54	0.55	0.52	0.42	18.13	15.12	18.10	12.75	41.41	37.24	39.63	30.75

yielding improper and inefficient placement of unpopular objects in nodes that can benefit from hosting popular objects with respect to their MCs demands. As a result, the distance traveled is increased and a node's $\mathcal{SC}$ is inefficiently utilized. However, in our scheme, replicas are assigned to equal-sized partitions and within each partition the best placement is searched among the partition members. This helps in distributing the traffic among the partitions to avoid the aforementioned problems. We note that in terms of packet collision, ML performs best since it is based on using the path that introduces the minimum loss rate by multiplying the success probability of each link along the path. This avoids lossy links and therefore, it reduces packet collisions. However, since ETT injects extra probes of data-packets, packet collision probability increases.

7.5.3 Average Throughput

In this subsection, we compare our schemes with the other heuristics with respect to the average throughput in the network. The results in Fig. 7.4 represent the average throughput distribution among different heuristics. We can notice that as the network size increases, the mean throughput decreases relatively among all the heuristics. This is an inevitable result as both distance and the number of served requests increase. It can be observed that *Lat-CDN* has a better throughput than the *Random* heuristic since servers cooperate to place objects

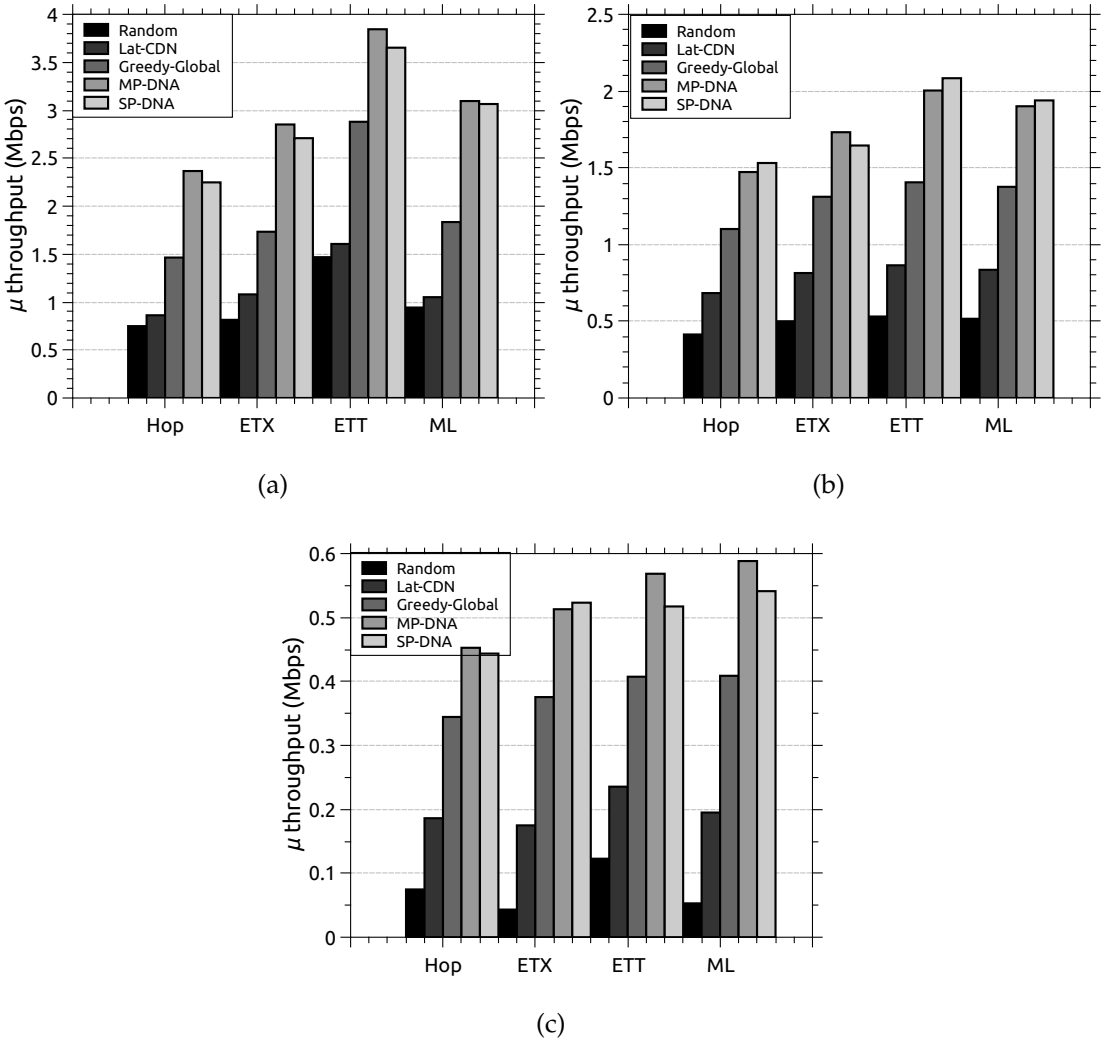


Figure 7.4: Average throughput (Mbps) for (a) $\mathcal{N} = 50$, (b) $\mathcal{N} = 150$ and (c) $\mathcal{N} = 300$.

with the high latency. However, it does not perform better than *Greedy-Global* and our schemes because it leads to improper replica placement as it considers latency cost without distinguishing the variable demand (popularity) among replica servers. Table 7.3 illustrates a comparison between our schemes and the other heuristics. We can notice that our schemes outperform the *Greedy-Global*. This is because *Greedy-Global* does not distinguish different sizes of objects as it may not be an issue for high-performance CDN servers. However, our schemes consider this crucial factor in finding the density share for each object, which yields to a significant performance improvement.

Our schemes also sort each group of object replicas in a decreasing order by size. By adopting this approach, a replica server can accommodate more objects of high popularity. In contrast; *Greedy-Global* is oblivious about the object size. This may not be an issue if we have abundant storage or when the objects have similar sizes. However, objects vary in size; therefore, our approach maximizes the benefit by prioritizing popular objects of large size over small ones. To compare with *Lat-CDN*, Table 7.3 clearly shows that our schemes have a significant performance gain over *Lat-CDN*. This gain is due to the same reason as in *Greedy-Global* (i.e., does not consider different sizes of object). Furthermore, it assumes similar popularity for an object among replica servers.

Table 7.3: Average throughput observed by the examined heuristics (Mbps).

	$\mathcal{N}$ = 50				$\mathcal{N}$ = 150				$\mathcal{N}$ = 300			
Metric	Hop	ETX	ETT	ML	Hop	ETX	ETT	ML	Hop	ETX	ETT	ML
Random	0.75	0.81	1.47	0.94	0.41	0.50	0.53	0.52	0.07	0.04	0.12	0.05
Lat-CDN	0.86	1.08	1.61	1.05	0.68	0.81	0.86	0.84	0.19	0.17	0.24	0.19
Greedy-Global	1.46	1.73	2.88	1.83	1.10	1.31	1.41	1.38	0.34	0.38	0.41	0.41
MP-DNA	2.37	2.85	3.84	3.10	1.47	1.73	2.00	1.90	0.45	0.51	0.57	0.59
SP-DNA	2.25	2.71	3.65	3.06	1.53	1.65	2.08	1.94	0.44	0.52	0.52	0.54

Although the *Greedy-Global* might lead to a better placement than our schemes

in extreme situations such as when the popularity is concentrated in one partition and comparatively low in the neighboring partitions. In such a situation, *Greedy-Global* might place multiple replicas in one partition; while our schemes place a replica in each partition. However, the traffic in such situation will be concentrated in that partition; therefore, two problems will arise. The first problem is the immense traffic within that partition, which will lead to problems like the contention to access the wireless medium, intra-flow and inter-flow interference, packet loss, congestion and degrading the link-quality. The second problem is the load imbalance between replica servers. The servers in the hotspot partition will be overloaded and the servers in the neighboring partitions are less loaded. Our scheme can achieve a better load balance than *Greedy-Global*, since requests can be forwarded to replica servers in the neighboring partitions to avoid the aforementioned problems. For *Greedy-Global*, this might not be a problem as long as there is sufficient bandwidth and high capacity CDN servers, but in the wireless environment these resources are scarce. We can also infer that our schemes perform best using the ETT metric, which selects paths with high link rates.

7.5.4 Average of No-retry Packets

In this subsection, we compare the schemes in terms of the number of packets transmitted at the link layer without retry. This reflects the impact of selecting high quality paths by avoiding lossy and congested links. We can notice from Fig. 7.5 that the average of sent packets without retry decreases with the network size. This is because more traffic is introduced, which will increase packet collisions. The *Random* scheme steeply drops because of the random placement of objects that increases the number of wireless transmissions. However, our schemes perform better than the *Greedy-Global*. This is due to the reduced

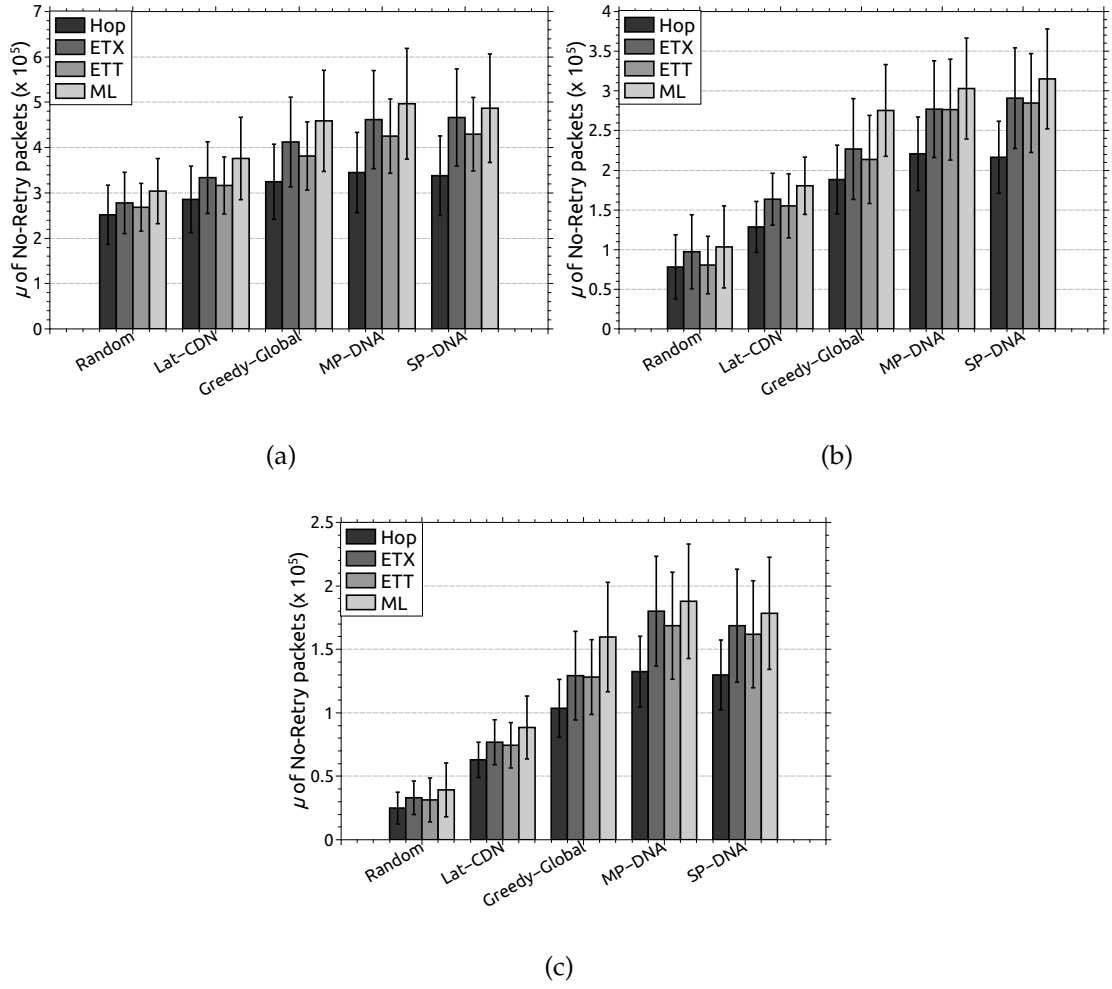


Figure 7.5: Average of No-Retry packets per node (x10^5) for (a) $\mathcal{N} = 50$, (b) $\mathcal{N} = 150$ and (c) $\mathcal{N} = 300$.

load on replica servers that avoids forwarding requests to hotspot zones where replicas are located. The *Lat-CDN* does not perform well because of the improper placement as explained previously. We can find out that in most cases the ML metric performs best.

7.5.5 Popularity Estimation vs. Throughput

In this subsection, we discuss the use of the popularity estimation model we used to approximate the number of replicas for the next replication period. It is up to the CM to log the previous popularity records of the served content. This functionality is not difficult since the CM periodically receives the popularity information from the DNs. The network size under investigation is $\mathcal{N} = 50$ and the ETT metric is used. Figs. 7.6a, 7.6c and 7.6e show the popularity estimation percentage based on the previous period, the predicted D-EWMA and the real percentage that is found at the end of the current period. This is to show how far the two estimations are from the real value. The three figures are based only on the top 100 ranked objects according to the real popularity value, where the top ranked object ID = 1. Note that each period has its different set of object ranks. This is because an object's popularity changes by time. Therefore, the top 100 object set might experience new objects added, current objects removed and an existing object's rank change within the range of 100. We can infer that when the values for the previous period are below the real values, this means that the popularity is increasing for this set of objects. When the situation is inversed, the popularity is decreasing. In our forecast model, which is based on the last two periods, We can notice that in most of the points (83%, 87% and 85% respectively) our estimation is closer to the real popularity than the previous popularity value. This is due to the trend consideration that uses the previous two real popularity values. However, in the false estimations where the

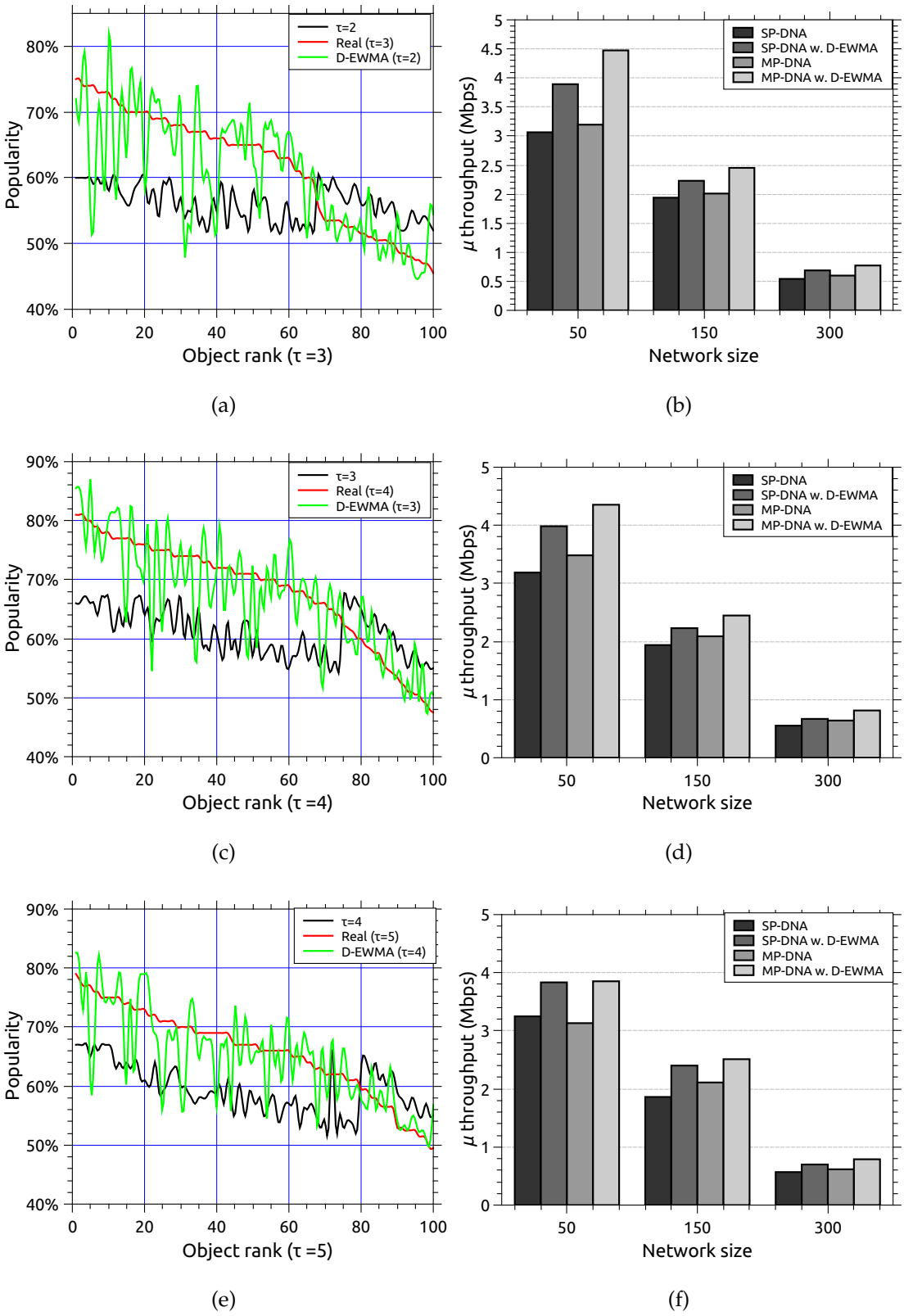


Figure 7.6: Average throughput improvement over 3 consecutive replication periods $\tau = 3, 4$ and 5.

value of the previous popularity is closer to the real values than our D-EWMA estimation. This is due to the unexpected concavity (up or down) of the clients' demands. The results in Figs. 7.6b, 7.6d and 7.6f provide an evidence on the performance gain in terms of the average throughput. We compare both of our schemes based on the previous popularity values and the predicted values using D-EWMA. Our obtained results are for the same periods ($\tau = 3$, 4 and 5). The throughput improvement ranges between 15% and 40%. This comes as a result of approximating objects popularity, which is critical for computing the number of replicas $p_m(\tau + 1)$ for the next period.

7.6 Summary

In this chapter, we addressed the shortcoming of using the simple hop-count in the cost function that we aim to optimize. To improve the access cost, other advanced link-quality metrics can be utilized. However, these metrics cannot be used directly, because they are designed for estimating the current link status to decide on the routing path that the packets will flow in. Since replication is costly and is intended for long period, we cannot use the instantaneous link value in our cost function as the link status fluctuates in a dynamic wireless environment. Therefore, our modified schemes collect statistical information about content popularity and the link-quality to assist the delegate nodes in the replica placement decision. Augmented with long-term link-quality metrics, a delegate node decides on the placement of replicas based on evaluating the trimmed mean for the wireless link between a node and its neighbors. This can avoid the outlier values that negatively affect the link estimation. Moreover, for replica server selection, clients' requests are directed to the closest replica server aided by the instantaneous link-quality metric. Another enhancement

to our schemes is the use of D-EWMA for popularity forecasting, which helps in approximating the number of replicas for the next replication period. Our simulation results show that these improvements can significantly enhance the network performance.

Chapter 8

Conclusions and Future Work

With the incremental deployment of infrastructure WMNs, popular Web content often suffers from delay and congestion due to large access demands and especially when there are a small number of Internet gateways. Strategically replicating popular content within the network is an effective approach to improve performance and achieve scalability. Content replication is a critical approach for WMNs, especially to bandwidth demanding objects like multimedia content downloading.

In this thesis, we demonstrate that support for replication at infrastructure nodes in WMNs (e.g., mesh routers) can significantly improve performance of mesh clients' perceived latency when accessing popular content from within the WMN instead of fetching it from the origin server. Our main contributions were two-fold. Firstly, we have considered the problem of content replication and placement in WMNs. We have shown that this problem in NP-complete for variable number of nodes $\mathcal{N}$ and variable number of object replicas p, therefore, we formulated the problem as a p-median problem. In a wireless environment, it is not efficient to rely on a central entity to find the different replica placements. To tackle this problem, we proposed two distributed schemes that exploit graph

partitioning such that in each partition, one of the p replicas is placed in a node that minimizes the total access cost according to the partition members. We have shown that the cooperation between mesh routers to disseminate the replicas, mitigates the access latency and traffic congestion in the network. We have also demonstrated that efficient placement can be constructed by exploiting the underlying network topology information available at mesh routers. We have shown that such support from mesh routers enables mesh clients to efficiently consume the limited network resources, reduce traffic interference in the network and increasing throughput at downloading nodes. Secondly, we enhance our schemes by:

- Considering the local popularity for an object within a partition such that if the object is not feasibly popular in a partition, then it will be forwarded to a larger partition, where a better placement can be considered.

- Showing that placement based on minimizing the simple hop-count does not yield to efficient placement, therefore, we proposed the utilization of advanced link-quality metrics to assist the placement decision by sampling the link metric during the replication period and then finding the trimmed mean link cost, which will be used in the placement decision for the next replication period. For replica server selection, a mesh router selects the server that has the shortest distance based on the instantaneous link metric used.

In this chapter, we conclude the thesis by highlighting the contributions in each chapter and summarizing the results. Also, we present a number of possible directions for future research.

8.1 Summary of Contributions

- In Chapter 2, we formulated the replica placement problem as a p-median problem in a P2P fashion, where a mesh router acts either as a server or as a client (on behalf of a mesh client). We have shown that when this problem is applied to a general network, the p-median problem can be difficult to solve to optimality since this class of problems is NP-complete and for large network size and large number of replicas, the problem cannot be solved in a polynomial time. Therefore, we use a heuristic approach to solve this problem.

- In Chapter 4, we proposed a hierarchical, scalable and distributed object replication and placement scheme. The scheme has two phases, the Network Setup Phase, which builds a balanced binary tree of multi-level partitions of the network. The balanced binary tree represents different levels of partition sizes ranging from coarse-grain to fine-grain. The Content Replication and Placement Phase is periodically performed by the delegate nodes and the content manager by using the hierarchy to collect the popularity information for different objects to compute the number of replicas needed for the next replication period, and then the placement job is assigned to delegate nodes. This converts the p-median problem into 1-median problem. The scheme takes into account the factors of object popularity and size to compute the number of replicas per object.

- In Chapter 5, we proposed a flat approach for content replication and placement. The scheme is flat, distributed and runs the placement in parallel. It aims to generate equal-sized partitions for different potential number of replicas, which can provide a better placement –compared to the hierarchical one– since the partitions have (semi)equal size, which reflects on

the reduction of access latency. The scheme is scalable since the p-median problem is down-sized to a 1-median problem, where each delegate node is responsible to place one replica of the p replicas. The obtained results show that the scheme can significantly improve network performance in terms of object access cost and computation/communication cost. However, it incurs extra computation and communication cost –compared to the hierarchical *SP-DNA* scheme– due to the multiple partitions' assignment for each delegate node. As a result, a delegate node will collect popularity statistics from various partitions' members yielding to extra computation/communication costs.

- In Chapter 6, we enhanced both of our schemes to consider the local popularity in a partition. This is suitable when the content popularity does not follow a uniform distribution. Two advantages can be achieved by this enhancement:a. When an object is not feasible to place in a partition, then it will be forwarded to a larger partition, where a feasible placement can be considered; and b. It avoids congested links and therefore, the network performance improves.The obtained results show that the proposed enhancements can significantly improve the network performance by reducing the object access cost. We have demonstrated that the enhancements introduce a slight increase in the convergence time and communication overhead due to the forwarding of infeasible objects to larger partitions.

- In Chapter 7, we proposed the exploitation of advanced link-quality metrics that reflect the channel conditions instead of using the hop-count metric in our objective function to improve the access cost. Due to the dynamic fluctuations in the channel capacity, we cannot rely on the instant value for the link-quality because replica placement is performed periodically. There-

fore, our modified schemes collect statistical information about content popularity and the link-quality to assist the delegate nodes in the replica placement decision. We use the trimmed mean value for the wireless link between a node and its neighbors to estimate the link cost. Based on the estimated long-term link cost, a delegate node decides on the placement of replicas. The instant link cost value is used for replica server selection when serving clients' requests. We also show that using the D-EWMA to model popularity forecasting, which helps in approximating the number of replicas for the next replication period. We used this model because popularity –in general– follows a trend. Although, this model is simple, but it shows an improved prediction than relying only on the previous popularity value.

8.2 Possible Future Works and Directions

In the following, some interesting research directions for extending the work presented in this thesis are introduced.

- **Heterogeneous node capacities**
  We have assumed that all mesh routers have the same upload bandwidth and storage in order to simplify our performance analysis and obtain useful insights about our proposed replication schemes. An interesting future research direction can be to relax these assumptions and consider more sophisticated replication schemes, where mesh routers have heterogeneous capacities.

- **Multi-source/Multi-path content retrieval**
  So far in this thesis, we only considered soliciting the content only from a single source (replica server). As a refinement to our schemes, when a MC

requests an object, the MR associated with the MC can select two (or may be more using some criteria) best sources that can serve the MC in parallel. This can be achieved by requesting a set of segments from each source. We anticipate that this can improve the latency time perceived by the MCs.

- **Enabling secure content replication in WMNs**
  One of the disadvantages of content replication in WMNs includes the spread of malware. This is an open problem. Solutions are orthogonal to what this thesis aims to tackle and need to be addressed.

Bibliography

[1] Roberto Canonico, Carmen Guerrero, and Andreas Mauthe. Content distribution infrastructures for community networks. *Computer Networks*, 53(4):431–433, 2009.

[2] MIT Roofnet. http://pdos.csail.mit.edu/roofnet/design/.

[3] TAP. http://taps.rice.edu/.

[4] Athens Wireless. http://www.awmn.net/.

[5] Berlin Freifunk. http://berlin.freifunk.net/.

[6] B.M. Maggs, F. Meyer auf der Heide, B. Vocking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 284 –293, oct 1997.

[7] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126 –134 vol.1, mar 1999.

[8] Saumitra M. Das, Himabindu Pucha, and Y. Charlie Hu. Mitigating the

gateway bottleneck via transparent cooperative caching in wireless mesh networks. *Ad Hoc Netw.*, 5:680–703, August 2007.

[9] Athena Vakali. Proxy cache replacement algorithms: A history-based approach. *World Wide Web*, 4(4):277–298, 2001.

[10] Tom M. Kroeger, Darrell D. E. Long, and Jeffrey C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[11] Akamai. http://www.akamai.com/.

[12] Limelight Networks. http://www.limelight.com//.

[13] Rajkumar Buyya, Mukaddim Pathan, and Athena Vakali, editors. *Content Delivery Networks (Lecture Notes Electrical Engineering)*. Springer-Verlag Gmbh, 1 edition, 2008.

[14] Jussi Kangasharju, Keith W Ross, and James W Roberts. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications Volume 24, N2 - 1 February 2000*, 02 2000.

[15] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, December 2004.

[16] Napster. http://www.napster.com/, 2006.

[17] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11:17–32, February 2003.

[18] Gnutella website. http://www.gnutella.com/, January 2007.

[19] Antonis Sidiropoulos, George Pallis, Dimitrios Katsaros, Konstantinos Stamos, Athena Vakali, and Yannis Manolopoulos. Prefetching in content distribution networks via web communities identification and outsourcing. *World Wide Web*, 11(1):39–70, 2008.

[20] Parth H. Pathak and Rudra Dutta. A survey of network design problems and joint design approaches in wireless mesh networks. *IEEE Communications Surveys and Tutorials*, 13(3):396–428, 2011.

[21] Thomas Plagemann, Roberto Canonico, Jordi Domingo-pascual, Carmen Guerrero, and Andreas Mauthe. Chapter 15 Infrastructures for Community Networks.

[22] Joo Paulo Barraca, Pedro Fernandes, Susana Sargento, and Rui M. Rocha. An architecture for community mesh networking. In *PIMRC*, pages 1–6. IEEE, 2008.

[23] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Comput. Netw.*, 47:445–487, March 2005.

[24] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wirel. Netw.*, 11(4):419–434, July 2005.

[25] Diego Passos, Célio de Albuquerque, Miguel Campista, Luís Costa, and Otto Duarte. Minimum loss multiplicative routing metrics for wireless mesh networks. *Journal of Internet Services and Applications*, 1:201–214, 2011. 10.1007/s13174-010-0015-6.

[26] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, MobiCom '04, pages 114–128, New York, NY, USA, 2004. ACM.

[27] Can Emre Koksal and Hari Balakrishnan. Quality-aware routing metrics for time-varying wireless mesh networks. *IEEE Journal on Selected Areas in Communications*, 24(11):1984–1994, 2006.

[28] Mohamed Karim Sbai, Chadi Barakat, Jaeyoung Choi, Anwar Al Hamra, and Thierry Turletti. Adapting bittorrent to wireless ad hoc networks. In *Proceedings of the 7th international conference on Ad-hoc, Mobile and Wireless Networks*, ADHOC-NOW '08, pages 189–203, Berlin, Heidelberg, 2008. Springer-Verlag.

[29] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 731–740, New York, NY, USA, 2002. ACM.

[30] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol (OLSR), 2003. Network Working Group Network Working Group.

[31] M.E.M. Campista, P.M. Esposito, I.M. Moraes, L.H.M. Costa, O.C.M. Duarte, D.G. Passos, C.V.N. de Albuquerque, D.C.M. Saade, and M.G. Rubinstein. Routing metrics and protocols for wireless mesh networks. *Network, IEEE*, 22(1):6 –12, jan.-feb. 2008.

[32] John Current, Mark Daskin, and David Schilling. Discrete network location models. *Industrial Engineering*, pages 81–118, 2002.

[33] R.Z. Farahani and M. Hekmatfar. *Facility Location: Concepts, Models, Algorithms and Case Studies*. Contributions to Management Science. Physica-Verlag HD, 2011.

[34] Andreas Klose and Andreas Drexl. Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29, 2005.

[35] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC '01, pages 21–29, New York, NY, USA, 2001. ACM.

[36] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[37] C. Canali, M.E. Renda, P. Santi, and S. Burresi. Enabling efficient peer-to-peer resource sharing in wireless mesh networks. *Mobile Computing, IEEE Transactions on*, 9(3):333 –347, march 2010.

[38] A. Al Asaad, S. Gopalakrishnan, and V. Leung. Peer-to-peer file sharing over wireless mesh networks. In *Communications, Computers and Signal Processing, 2009. PacRim 2009. IEEE Pacific Rim Conference on*, pages 697 –702, aug. 2009.

[39] S.M. ElRakabawy and C. Lindemann. Peer-to-peer file transfer in wireless mesh networks. In *Wireless on Demand Network Systems and Services, 2007. WONS '07. Fourth Annual Conference on*, pages 114–121, 2007.

[40] Shahram Ghandeharizadeh, Tooraj Helmi, Taehee Jung, Shyam Kapadia, and Shahin Shayandeh. An evaluation of two policies for simple placement of continuous media in multi-hop wireless networks. In *Proceedings of the twelfth international conference on distributed multimedia systems*, DMS '06, 2006.

[41] Jen-Wen Ding, Wan-Ting Wang, and Chu-Fu Wang. An efficient data replication scheme for peer-to-peer video streaming over wireless-mesh community networks. In *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP '08 International Conference on*, pages 767 –770, aug. 2008.

[42] Pavan Nuggehalli, Vikram Srinivasan, Carla-Fabiana Chiasserini, and Ramesh R. Rao. Efficient cache placement in multi-hop wireless networks. *IEEE/ACM Trans. Netw.*, 14:1045–1055, October 2006.

[43] Pawel Winter. Steiner problem in networks: A survey. *ACM Networks*, 17(2):129–167, 1987.

[44] Heesu Im, Yugyung Lee, and Saewoong Bahk. Incentive-driven content distribution in wireless multimedia service networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1 –5, dec. 2010.

[45] Savvas Gitzenis, Georgios S. Paschos, and Leandros Tassiulas. Asymptotic laws for content replication and delivery in wireless networks. In Albert G. Greenberg and Kazem Sohraby, editors, *INFOCOM*, pages 531–539. IEEE, 2012.

[46] Antonio Corradi and Eugenio Magistretti. Comparing and evaluating lightweight solutions for replica dissemination and retrieval in dense

manets. In *Proceedings of the 10th IEEE Symposium on Computers and Communications*, ISCC '05, pages 43–50, Washington, DC, USA, 2005. IEEE Computer Society.

[47] Bong-Jun Ko and Dan Rubenstein. Distributed self-stabilizing placement of replicated resources in emerging networks. *IEEE/ACM Trans. Netw.*, 13:476–487, June 2005.

[48] C. Casetti, C. Chiasserini, M. Fiore, Chi-Anh La, and P. Michiardi. P2p cache-and-forward mechanisms for mobile ad hoc networks. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 386–392, 2009.

[49] C.-A. La, Pietro Michiardi, Claudio Casetti, Carla-Fabiana Chiasserini, and Marco Fiore. A lightweight distributed solution to content replication in mobile networks. In *WCNC*, pages 1–6, 2010.

[50] Panagiotis Pantazopoulos, Ioannis Stavrakakis, Andrea Passarella, and Marco Conti. Efficient social-aware content placement in opportunistic networks. In *Proceedings of the 7th international conference on Wireless on-demand network systems and services*, WONS'10, pages 17–24, Piscataway, NJ, USA, 2010. IEEE Press.

[51] Mohamed Karim Sbai, Emna Salhi, and Chadi Barakat. P2p content sharing in spontaneous multi-hop wireless networks. In *Proceedings of the 2nd international conference on COMmunication systems and NETworks*, COMSNETS'10, pages 203–212, Piscataway, NJ, USA, 2010. IEEE Press.

[52] E. Atsan, D. Altinbuken, and O. Ozkasap. Scalar data replication performance in mobile ad hoc applications. In *Computer and Information Sciences,*

2009. ISCIS 2009. 24th International Symposium on, pages 369 –374, sept. 2009.

[53] P. Bellavista, A. Corradi, and E. Magistretti. Lightweight replication middleware for data and service components in dense manets. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pages 142 – 152, june 2005.

[54] K. Shi, R. Chen, and H. Jin. Zone-based replication scheme for mobile ad hoc networks using cross-layer design. *Mobile Ad-hoc and Sensor Networks*, pages 698–710, 2006.

[55] Chi Anh La, Pietro Michiardi, Claudio E Casetti, Carla-Fabiana Chiasserini, and Marco Fiore. Content Replication in Mobile Networks. *IEEE Journal on Selected Areas in Communications, Special Issue Cooperative Networking Challenges and Applications, Volume 30, N9, October 2012*, 12 2011.

[56] N. Chand, R.C. Joshi, and M. Misra. Efficient cooperative caching in ad hoc networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–8, 2006.

[57] Joonho Cho, Seungtaek Oh, Jaemyoung Kim, Hyeong-Ho Lee, and Joonwon Lee. Neighbor caching in multi-hop wireless ad hoc networks. *Communications Letters, IEEE*, 7(11):525–527, 2003.

[58] M. Amadeo and A. Molinaro. Chanet: A content-centric architecture for ieee 802.11 manets. In *Network of the Future (NOF), 2011 International Conference on the*, pages 122 –127, nov. 2011.

[59] P. Mitra and C. Poellabauer. Service sharing in mobile sensing systems. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pages 110 –114, dec. 2011.

[60] Yi-Wei Ting and Yeim-Kuan Chang. A novel cooperative caching scheme for wireless ad hoc networks: Groupcaching. *Networking, Architecture, and Storage, International Conference on*, 0:62–68, 2007.

[61] Yu Du and Sandeep K. S. Gupta. Coop - a cooperative caching service in manets. In *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, ICAS-ICNS '05, pages 58–, Washington, DC, USA, 2005. IEEE Computer Society.

[62] Hassan Artail, Haidar Safa, Khaleel Mershad, Zahy Abou-Atme, and Nabeel Sulieman. Coacs: A cooperative and adaptive caching system for manets. *IEEE Transactions on Mobile Computing*, 7(8):961–977, August 2008.

[63] Shruti Sanadhya, Raghupathy Sivakumar, Kyu-Han Kim, Paul Congdon, Sriram Lakshmanan, and Jatinder Pal Singh. Asymmetric caching: improved network deduplication for mobile devices. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, Mobicom '12, pages 161–172, New York, NY, USA, 2012. ACM.

[64] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.

[65] Jing Zhao, Ping Zhang, Guohong Cao, and C. R. Das. Cooperative caching in wireless p2p networks: Design, implementation, and evaluation. *IEEE Trans. Parallel Distrib. Syst.*, 21(2):229–241, February 2010.

[66] Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and K.K. Ramakrishnan. Coexist: a hybrid approach for content oriented publish/subscribe systems. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ICN '12, pages 31–36, New York, NY, USA, 2012. ACM.

[67] Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and K.K. Ramakrishnan. G-copss: A content centric communication infrastructure for gaming applications. *2012 IEEE 32nd International Conference on Distributed Computing Systems*, 0:355–365, 2012.

[68] Fahad R. Dogar, Amar Phanishayee, Himabindu Pucha, Olatunji Ruwase, and David G. Andersen. Ditto: a system for opportunistic caching in multi-hop wireless networks. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, MobiCom '08, pages 279–290, New York, NY, USA, 2008. ACM.

[69] Shudong Jin and Limin Wang. Content and service replication strategies in multi-hop wireless mesh networks. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '05, pages 79–86, New York, NY, USA, 2005. ACM.

[70] Pei Cao and Sandy Irani. Cost-aware www proxy caching algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, pages 18–18, Berkeley, CA, USA, 1997. USENIX Association.

[71] Yingnan Zhu, Wenjun Zeng, Hang Liu, Yang Guo, and Saurabh Mathur. Supporting video streaming services in infrastructure wireless mesh networks: Architecture and protocols. In *ICC*, pages 1850–1855, 2008.

[72] Thabo K. R. Nkwe, Mieso K. Denko, and Jason B. Ernst. Data ubiquity in autonomic wireless mesh networks. *J. Ambient Intelligence and Humanized Computing*, 1(1):3–13, 2010.

[73] Enhua Tan, Lei Guo, Songqing Chen, and Xiaodong Zhang. Scap: Smart caching inwireless access points to improve p2p streaming. In *Dis-*

tributed Computing Systems, 2007. ICDCS '07. 27th International Conference on, page 61, june 2007.

[74] Yingnan Zhu, Hang Liu, Yang Guo, and Wenjun Zeng. Network assisted media streaming in multi-hop wireless networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1 –7, 31 2011-aug. 4 2011.

[75] Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Replication for web hosting systems. *ACM Comput. Surv.*, 36(3):291–334, September 2004.

[76] Bo Li, Mordecai J. Golin, Giuseppe F. Italiano, Xin Deng, and Kazem Sohraby. On the optimal placement of web proxies in the internet. In *INFOCOM*, pages 1282–1290, 1999.

[77] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the placement of web server replicas. In *INFOCOM*, pages 1587–1596, 2001.

[78] P. Krishnan, Danny Raz, and Yuval Shavitt. The cache location problem. *IEEE/ACM Trans. Netw.*, 8(5):568–582, October 2000.

[79] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. On the placement of internet instrumentation. In *INFOCOM*, pages 295–304, 2000.

[80] Magnus Karlsson and Mallik Mahalingam. Do we need replica placement algorithms in content delivery networks. In *In Proceedings of the International Workshop on Web Content Caching and Distribution (WCW)*, pages 117–128, 2002.

[81] Pavlin Radoslavov, Ramesh Govindan, and Deborah Estrin. Topology-informed internet replica placement. *Computer Communications*, 25(4):384–392, 2002.

[82] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

[83] Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):376 – 383, 2002.

[84] George Pallis, Athena Vakali, Konstantinos Stamos, Antonis Sidiropoulos, Dimitrios Katsaros, and Yannis Manolopoulos. A latency-based object placement approach in content distribution networks. In *Proceedings of the Third Latin American Web Congress*, pages 140–, Washington, DC, USA, 2005. IEEE Computer Society.

[85] Jian Zhou, Xin Zhang, Laxmi N. Bhuyan, and Bin Liu. Clustered k-center: Effective replica placement in peer-to-peer systems. In *GLOBECOM*, pages 2008–2013, 2007.

[86] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. In *Innovations in NGN: Future Network and Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference*, pages 367–374, 2008.

[87] K. Selçuk Candan and Nikhil Iyer. Server replication in interactive, push-

based data delivery networks. In Maria Luisa Sapino and Prashant J. Shenoy, editors, *Multimedia Information Systems*, pages 50–59, 2004.

[88] Jian Ni, D.H.K. Tsang, I.S.H. Yeung, and Xiaojun Hei. Hierarchical content routing in large-scale multimedia content delivery network. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 2, pages 854 – 859 vol.2, may 2003.

[89] H.M.N.D. Bandara and A.P. Jayasumana. Exploiting communities for enhancing lookup performance in structured p2p systems. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –6, june 2011.

[90] Guofu Feng, Wenzhong Li, Sanglu Lu, and Daoxu Chen. The optimal replica distribution to minimize the search size in the unstructured overlay. In *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, pages 591 –596, sept. 2010.

[91] B W Kernighan and S Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[92] George Karypis and Vipin Kumar. Metis - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995.

[93] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. *Data Engineering, International Conference on*, 0:49, 2003.

[94] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[95] Alfonso Ariza Quintana, Eduardo Casilari, and Alicia Triviño. Implementation of manet routing protocols on omnet++, 2008.

[96] I. Baumgart, B. Heep, and S. Krause. Oversim: A flexible overlay network simulation framework. In *IEEE Global Internet Symposium, 2007*, pages 79–84, 2007.

[97] Venkata N. Padmanabhan and Lili Qiu. The content and access dynamics of a busy web site: findings and implications. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 111–123, New York, NY, USA, 2000. ACM.

[98] Yih-Farn Chen, Yennun Huang, Rittwik Jana, Hongbo Jiang, Michael Rabinovich, Jeremy Rahe, Bin Wei, and Zhen Xiao. Towards capacity and profit optimization of video-on-demand services in a peer-assisted iptv platform. *Multimedia Syst.*, 15(1):19–32, 2009.

[99] Carlos Cunha, Azer Bestavros, and Mark Crovella. Characteristics of www client-based traces. Technical report, Boston, MA, USA, 1995.

[100] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. pages 62–68, 2001.

[101] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.

[102] Guohua Zhang, Yiyu Wu, and Yonghe Liu. Stability and sensitivity for congestion control in wireless mesh networks with time varying link capacities. *Ad Hoc Netw.*, 5(6):769–785, August 2007.

[103] AH Welsh. The trimmed mean in the linear model. *The Annals of Statistics*, 15(1):20–36, 1987.

[104] Michael Rabinovich and Oliver Spatschek. *Web caching and replication*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[105] Swaminathan Sivasubramanian, Guillaume Pierre, Maarten van Steen, and Gustavo Alonso. Analysis of caching and replication strategies for web applications. *IEEE Internet Computing*, 11(1):60–66, 2007.

[106] Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

[107] Joseph J. LaViola. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In *Proceedings of the workshop on Virtual environments 2003*, EGVE '03, pages 199–206, New York, NY, USA, 2003. ACM.

[108] Siddharth Mitra, Mayank Agrawal, Amit Yadav, Niklas Carlsson, Derek Eager, and Anirban Mahanti. Characterizing web-based video sharing workloads. *ACM Trans. Web*, 5(2):8:1–8:27, May 2011.