REDES ORIENTADAS À INFORMAÇÃO COM ESTRATÉGIAS PROBABILÍSTICAS ACOPLADAS PARA BUSCA E REPLICAÇÃO

Guilherme de Melo Baptista Domingues

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Edmundo Alburquerque de Souza e Silva
Rosa Maria Meri Leão
Daniel Sadoc Menasché

Rio de Janeiro
Outubro de 2015

REDES ORIENTADAS À INFORMAÇÃO COM ESTRATÉGIAS PROBABILÍSTICAS ACOPLADAS PARA BUSCA E REPLICAÇÃO

Guilherme de Melo Baptista Domingues

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Edmundo Alburquerque de Souza e Silva, Ph.D.

Prof. Daniel Sadoc Menasché, Ph.D.

Prof. Artur Ziviani, Dr.

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Pedro Braconnot Velloso, Dr.

RIO DE JANEIRO, RJ – BRASIL

OUTUBRO DE 2015

Domingues, Guilherme de Melo Baptista

Redes Orientadas à Informação com Estratégias Probabilísticas Acopladas para Busca e Replicação/Guilherme de Melo Baptista Domingues. – Rio de Janeiro: UFRJ/COPPE, 2015.

XII, 64 p.: il.; 29, 7cm.

Orientadores: Edmundo Alburquerque de Souza e Silva
Rosa Maria Meri Leão
Daniel Sadoc Menasché

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 58 – 64.

1. Information Centric Networks. 2. Computer Networks. 3. Reliability Theory. I. Silva, Edmundo Alburquerque de Souza e *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha família pelo dom da vida e pelo amparo ao longo desses anos.

Aos professores, alunos e equipe do LAND.

Agradecimentos

Agradeço à minha família, pelo apoio e consideração para comigo ao longo de toda a tese, em especial a meu pai Carlos, à minha mãe Lina e à minha irmã Inês, sem os quais essa tese não seria possível. Agradeço aos meus orientadores de tese, Professores Edmundo Albuquerque de Souza e Silva, Rosa Maria Meri Leão e Daniel Sadoc Menasché que contribuíram com uma grandeza difícil de mensurar em palavras, em seus conhecimentos técnicos científicos e sobretudo em seu acolhimento afetivo, sem os quais essa tese não teria sdo possível. Agradeço à Carolina por todos os momentos de escuta durante a composição deste trabalho, portadora de um brilhantismo singular ao prover palavras sábias e de muito conforto sem a qual essa tese não teria sido possível. Agradeço a todos os professores e alunos do Land com os quais tive a honra de conviver, bem como do PESC, cabendo um agradecimento especial ao Gaspare, com quem pude compartilhar toda minha trajetória desde que começamos nosso doutorado juntos, e que sempre esteve ao meu lado em toda minha jornada, sem os quais essa tese não seria possível. Por fim, agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro, sem o qual essa tese não teria sido possível.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

REDES ORIENTADAS À INFORMAÇÃO COM ESTRATÉGIAS PROBABILÍSTICAS ACOPLADAS PARA BUSCA E REPLICAÇÃO

Guilherme de Melo Baptista Domingues

Outubro/2015

Orientadores: Edmundo Alburquerque de Souza e Silva
Rosa Maria Meri Leão
Daniel Sadoc Menasché

Programa: Engenharia de Sistemas e Computação

Redes orientadas à informação estão em evidência atualmente. Essas redes são responsáveis pela replicação e alocação de conteúdos, que são produzidos em grande quantidade pelos usuários. Em redes orientadas à informação, os conteúdos se tornam a essência. Desafios de escalabilidade e confiabilidade em cenários de redes orientadas à conteúdos motivaram o desenho de uma nova arquitetura, bem como os modelos analíticos e a análise de métricas de desempenho nessa tese.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

INFORMATION CENTRIC NETWORKS WITH COUPLED PROBABILISTIC SEARCH AND CACHING POLICIES

Guilherme de Melo Baptista Domingues

October/2015

Advisors: Edmundo Alburquerque de Souza e Silva
Rosa Maria Meri Leão
Daniel Sadoc Menasché

Department: Systems Engineering and Computer Science

Information centric networks are in evidence nowadays. These networks should replicate and distribute efficiently the content produced by users. In ICNs, named data is the focus of the network. Scalability and reliability issues in information centric networks scenario motivated the new architecture design, the corresponding analytical model and the analysis of performance metrics in this thesis.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introduction

The Internet was initially conceived as an infrastructure for host-to-host communications, with packets being forwarded along network nodes solely based on destination host addresses. File transfer, email exchange and remote access comprise examples of key applications tailored for this infrastructure [1]. Nonetheless, in recent years, the Internet focus has been changing towards content dissemination (Youtube [2], Wikipedia [3], Netflix [4], VideoLectures RIO [5]).

As a first attempt to adapt the Internet to a content dissemination network, overlay networks, such as content distribution networks (CDNs) and peer-to-peer networks (P2P) have been proposed. A CDN ([6], [7]) is a distributed system aiming to route content requests to hosts at ISPs close to users at the edge of the network, in order to minimize the total delay cost experienced by requesters while offloading custodians (origin servers) at publisher's networks (see Figure 1.1). A P2P network ([8], [9]) is a distributed system where peers act as clients and servers (see Figure 1.2) simultaneously. Peers interested in the same content exchange content chunks among themselves, in a coordinate fashion.

In CDNs, replicas of popular contents become stored at servers in different Internet service providers (ISPs). Those servers operate as content caches, placed at the edge of the network. Content distribution policy requires capacity planning by CDN owners. Users send requests to the CDN, wherein algorithms, operating as central controllers, redirect the ingress traffic to servers storing the requested contents. Those algorithms need to be aware of all replica distribution across different ISP servers. Therefore, all policies for traffic exchange between autonomous systems

in the Internet must be accounted by these central controllers.

In native P2P networks, the absence of incentive policies for peers to stay in the swarm, after gathering all content requested, may impose scalability issues [10]. In case a peer services others in a swarm and this peer leaves the swarm, the overall service capacity for the content being served by the swarm is impacted. Also, when a peer leaves a swarm, some content chunks may remain absent so that the content itself cannot be retrieved entirely by the remaining peers. Hybrid P2P networks try to enhance delivery performance by considering persistent hosts at the network.

According to [11], global IP traffic will reach a zettabyte era by 2017. Video will correspond to approximately 80 to 90 percentage of global consumer traffic. In CDNs and P2P networks, it is not easy to optimize content delivery performance at large scale scenarios. For overlay networks, content distribution depend solely on the information present at the overlay topology, regardless of information at the underlying network. This has contributed to many inefficiencies [12].

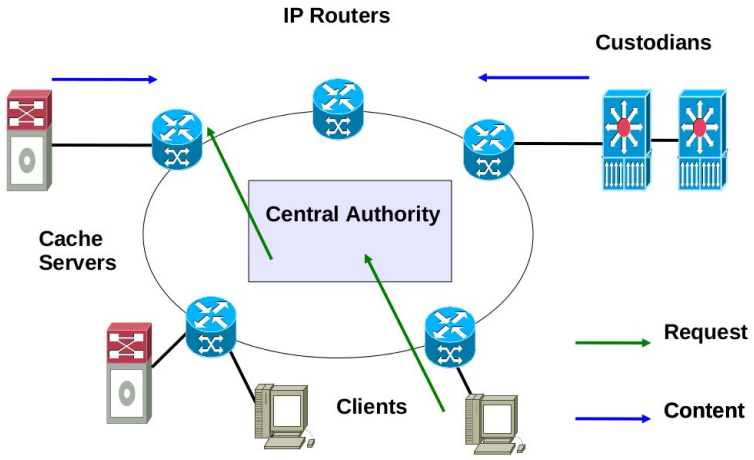


Figura 1.1: CDN - Content Distribution Networks

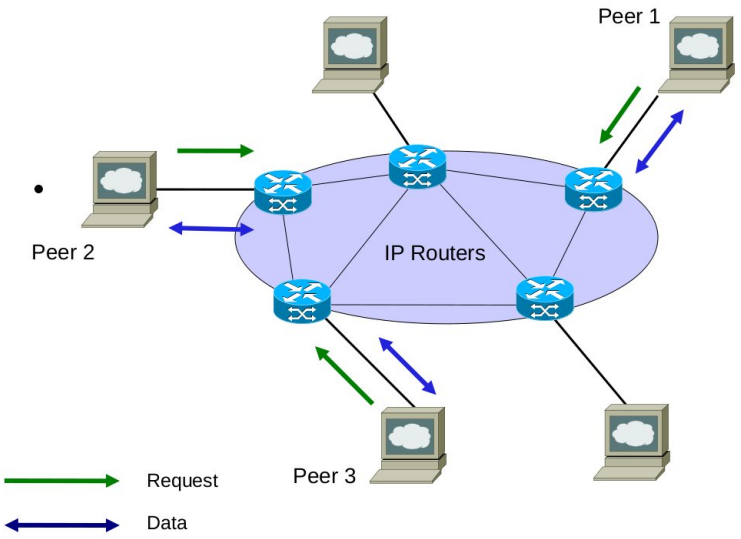


Figura 1.2: P2P - Peer to Peer Networks

For instance, content replicas may be stored in caches at network level, enhancing performance for content retrieval. In Figure 1.3, a requested content gets stored at the publisher network, at an origin server. A connection is established between user A (requester) and the origin server (destination) from where content is retrieved by user A. Due to previous requests by users B and C to the same origin server, in case A could get supplied by caches within network nodes close to B and C, the total cost for A to receive the content would be expected to be reduced.

Therefore. a new network design for large scale scenarios, accounting for the underlying network topology and information flow, towards the improvement of content delivery efficiency and content availability has emerged [13], [14]. The research community has addressed a paradigm shift, wherein content delivery becomes a native network feature: a paradigm shift towards a information centric network (ICN). The key idea behind ICNs comes from the evidence that large amounts of content, once produced, become replicated many times without the network getting involved in setting the distribution policies. In ICNs, each content is associated to a name, what allows contents to be searched, cached and replicated according to native network layer policies (see Figure 1.4).

In ICNs, once a request for a content is generated it flows towards the custodian through the content (or cache) routers. Each (content) router has a cache that

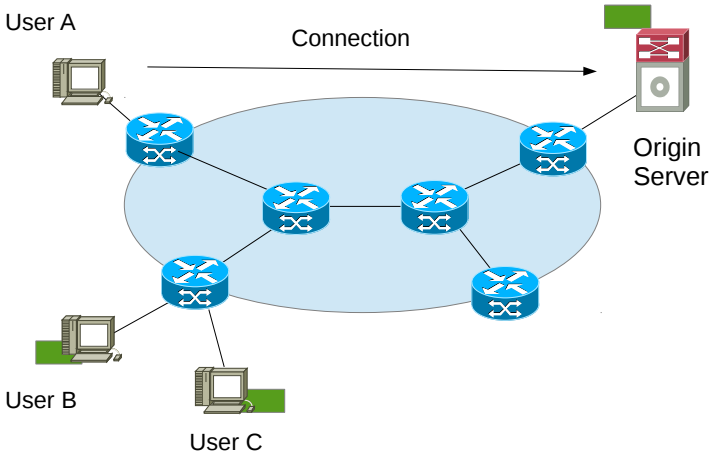


Figura 1.3: Content Retrieval - Native IP Network

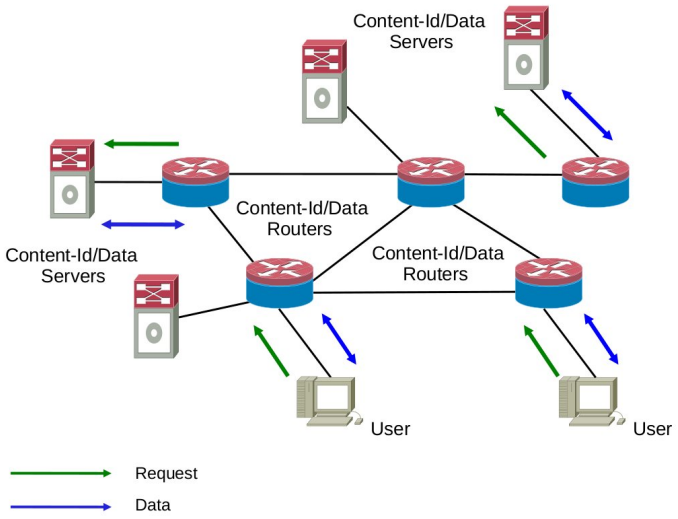


Figura 1.4: Information Centric Network

can store the contents in addition to the custodians. Therefore, a request that finds the content stored in a router does not have to access the custodian. This potentially alleviates the load at the custodians, reduces the delay to retrieve the content and the overall traffic in the network. The efficient management of the distributed storage resources in the network coupled with the routing of requests for information retrieval are issues of fundamental importance [15] in ICNs. However, there is an urge to study search and placement problems under a holistic perspective.

Our contributions in this work rely on a new architecture to circumvent ICN open challenges [16] for architecture design, regarding request forwarding and content placement in network nodes, under a self-adaptive architecture to cope with environment changes. These architecture design contributions are depicted at the end of Chapter 2. We provide analytical models for the entire architecture. Our quantitative analysis contributes to the evaluation of the interplay of (i) random walks (RW) as the basis for our content search policy, exploring the vicinity of caches the requests cross while being delivered to custodians, and (ii) reinforced counters (RC), as a set of independent time counters as the basis for our content placement policy. Through our analytical models, performance metrics trade-offs can be evaluated. Optimal values for the performance metrics can be obtained when constraints such as maximum cache size and maximum cache writing rates are considered.

In Chapter 2 of this thesis, we provide more details about information centric networks (ICNs). We describe key architectures in the literature and open issues regarding ICNs. In Chapter 3 we describe the ICN architecture proposed in this thesis. In Chapter 4 we analyze local cache performance. Chapters 5 and 6 present mathematical models for the architecture with respective analysis. Finally, in Chapter 7, we present our conclusions.

Capítulo 2

Information Centric Networks - ICNs

The introduction of scalable content distribution applications over the internet packet switched architecture has driven a research effort to investigate a clean-slate architectural approach for the Internet. Information centric networks (ICNs) have been proposed as a new approach for the Future Internet [15], [17], [18]. The named data scheme adopted impacts the routing policy. Each content has a unique name. Nonetheless, names can be aggregated by common prefixes, related to the publisher that issued the content to the network.

A set of network entities routes the content requests up to a replica of the desired content stored at the network. This replica can be stored either in custodians or at caches distributed along the network. The request routing policy guides a request within the network, such that a request ends up finding a replica of the searched content. In case the replica reaches a custodian, cache misses should have occurred. Otherwise, we say that a cache hit takes place, offloading the custodians.

The content routing policy guides a content within the network. In case of a cache miss, content is send from custodians to users. In case of a cache hit, content is send directly from caches to users. Request routing is related to content routing in the sense that request routing aims to balance the load for a given content, among several repositories this content is stored at [19], [20]. Figure 2.1 shows some of the most important ICN features.

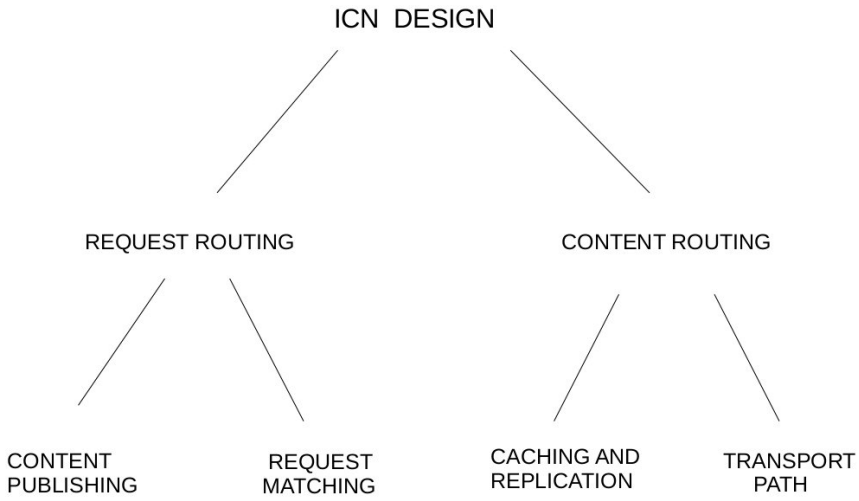


Figura 2.1: ICN Design Features

Request Matching: Users issue requests to the network. Those requests are routed until a matching between a request and a corresponding stored content takes place within the network;

Content Publishing: Each content is published into the network without any explicit destination address, but structured according to a *name* criteria;

Caching and Replication: Caching can be potentially implemented by all routers, which turns caching pervasive along the entire network. Caching is done on the network level.

Transport Path: Content can be send to users following a reverse path left as a trail by requests forwarded to contents or throughout an IP network.

In coupled ICNs (see Figure 2.2), the nodes in charge for content routing and request routing are the same. If separate networks are used instead, request routing gets decoupled from content routing (see Figure 2.3). In ICNs, content is able to be directly forwarded to requesters from the network itself. ICNs design targets keeping up content delivery performance in light of future scalability, robustness

and reliability issues.

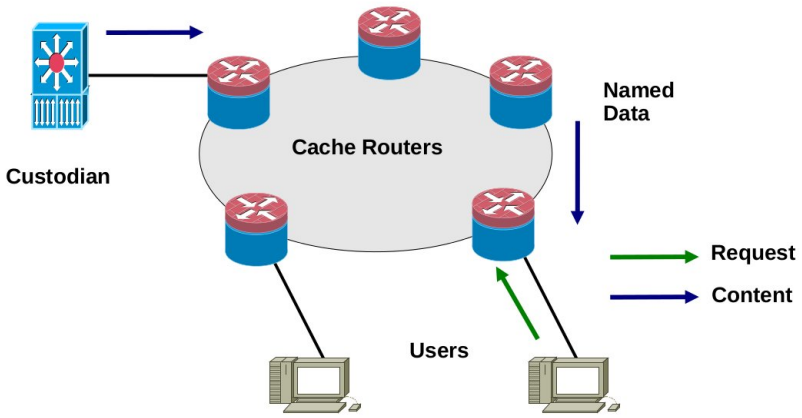


Figura 2.2: Coupled ICN Approach

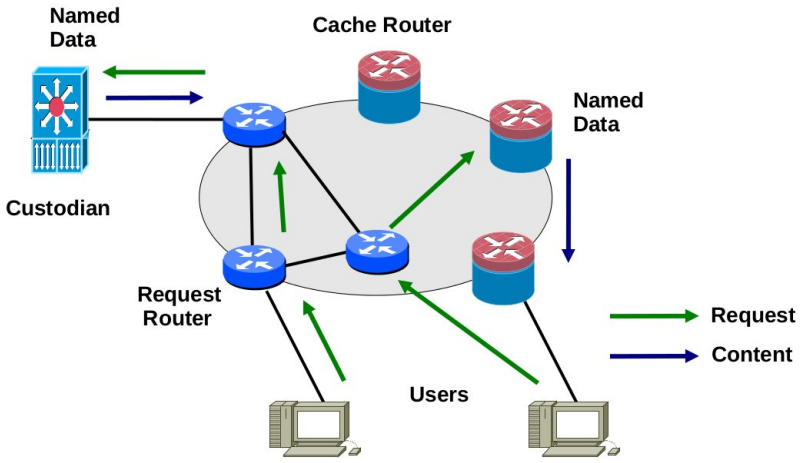


Figura 2.3: Decoupled ICN Approach

In ICNs, the network acquires full potential for registering all requests and content flows. This allows the design of optimal strategies for massive content distribution. A match between requests and content itself takes place at network level. Transparent caching acts on behalf of requesters and content caching policies do not require centralized algorithms, such as in CDNs. Local policies can enforce caching within network nodes.

Both content requesters and publishers become unaware of cached content, as this knowledge remains within the network. Therefore, transparent caching policy is adopted. Users can retrieve content from storage locations geographically nearby. Declining costs for memory opens the door for all routers being deployed with caches, capable of directly forwarding contents to users. A pervasive storage strategy aims to decrease transmission traffic cost, increasing the speed of response. Named contents allow cached content to be easily retrieved, compared to costly techniques like deep packet inspection [21], [22], in case named content is not considered.

2.1 Dona Architecture

Data Oriented Network Architecture (DONA [23]) architecture was one of the first complete ICN architectures attempts. Instead of URLs binding content names to specific locations through DNS, persistent flat names were considered instead. With persistent names, contents were able to be cached and replicated at network level, being requested by names from users, increasing information availability.

Names in DONA are associated with the corresponding publisher, with unique identification across the entire network. A name resolution system operates on top of an IP network. DONA architecture comprises a set of interconnected nodes named resolution handlers (RH), with at least one RH per autonomous system (AS) (see Figure 2.4). A RH in a AS gets responsible for registering published contents at this AS. The set of interconnected RHs composes a name resolution system (NRS), wherein a mechanism maps flat names to IP locators.

After a content has been published in a AS, the corresponding RH receiving the publishing registration at the AS propagates this registration to the RHs in its

parent and peering AS. Those RHs store a pointer to the origin RH, for a registered published content. Content registrations get replicated across the network, up to the RH at tier-1 AS. Henceforth, the RH at tier-1 AS becomes aware of all published contents registrations. Content requesters are assumed to learn a content name through some trusted external mechanism (e.g., a search engine). Requests are send to local RHs and forwarded by the NRS towards a content storage location, according to the pointers stored at the NRS.

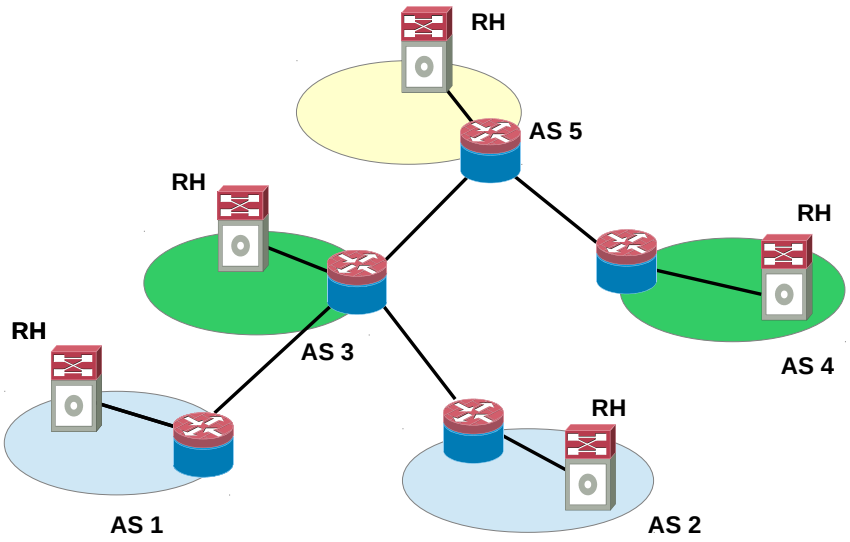


Figura 2.4: DONA Architecture

Content may be send to a requester throughout the underlying IP network, under a decoupled policy, or following the path of a request since the origin to the destination, in a reverse order. Caching may be supported by the RHs. Each RH is allowed to decide if a content should be cached at the AS the RH belongs to. To this aim, an RH is able to replace the IP source address of a request. In this case, this RH will be considered the origin of a request from the destination perspective. Content will be first delivered to the origin RH, before being send to a requester. The new content location is announced to RHs at parent and peering AS.

Figure 2.5 shows a searched content published at AS4, being cached at AS2

and delivered to AS1. Content request flow through RHs, reaching AS4 from AS1. DONA requires name resolution state at RHs increase, as the tier level increases. Therefore, the top-level resolution servers must store huge amounts of data, where information about all published content is replicated entirely at the top level RHs.

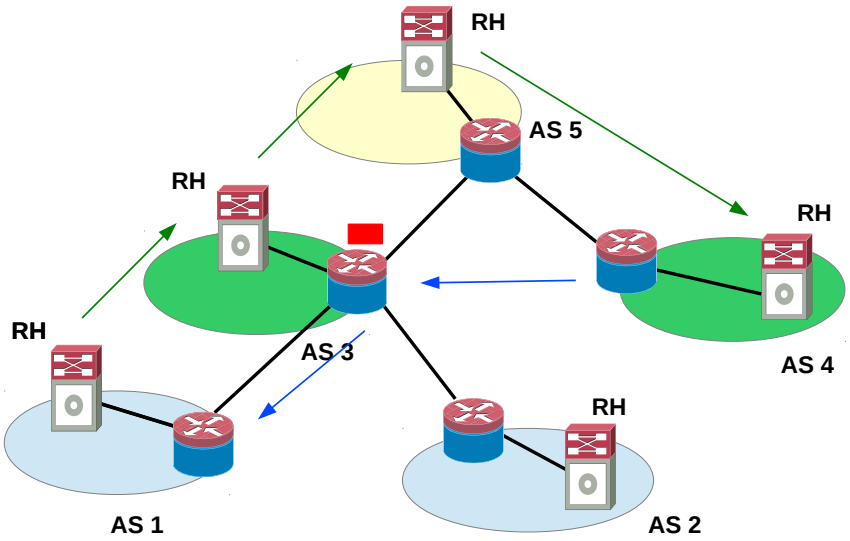


Figura 2.5: Request / Content Flow - DONA

2.2 DHT Architectures

The Publish Subscribe Internet Routing Paradigm architecture (PSIRP [24]/ PURSUIT project [25] architecture and Netinf [26]/ SAIL project [27] are proposals that handle name resolution by a set of routing nodes (RNs), disposed along a network implemented as an hierarchical DHT [28].

When a publisher issues a request to a local RN, this request is routed by the DHT to a destination RN, responsible for the corresponding DHT scope of the request. All requests for the same content end up being sent to the same destination RN. A match between a request and a published content takes place at the destination RN, through locators. In PSIRP, after the match, a topology manager

(TM) node gets instructed to establish a connection between the publisher and the subscriber for content delivery.

Content is sent across a set of forwarding nodes (FNs) using the bloom filter technique. In Netinf, contents are routed to subscribers through FN's belonging to an IP network. In both PSIRP and NetInf, caching is supported by FNs, with FNs being able to advertise cached information to RNs, in order to enhance search efficiency. In Netinf, local NRS can be deployed along the hierarchy. In this case, at the top level, a global NRS (GRS) [29], [30] is responsible for interconnecting the local NRS.

Figure 2.6 illustrates the baselines of DHT proposals. In hierarchical DHTs, a flexible approach able to capture the dynamic routing relationships between AS's, is still missing with routing efficiency for this strategy presenting several drawbacks [31]. Also, bloom filters do not scale to Internet sizes and presents higher number of false positives for content delivery, as more links are added to the network [32].

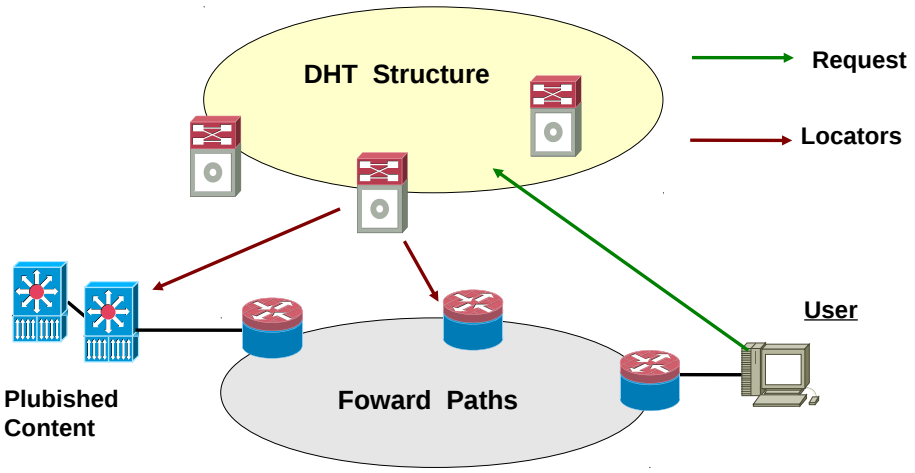


Figura 2.6: DHT Proposals

2.3 CCN architecture

CCN [33] architecture handles name resolution through content routing tables across routers. Names in CCN are hierarchical and may be similar to URLs, able to be aggregated under prefixes. Requests flows across content routers (CR), each CR maintaining three data structures: the Forwarding Information Base (FIB), the Pending Interest Table (PIT) and the Content Store (CS). Figure 2.7 illustrates this proposal with names aggregated according to second level DNS names.

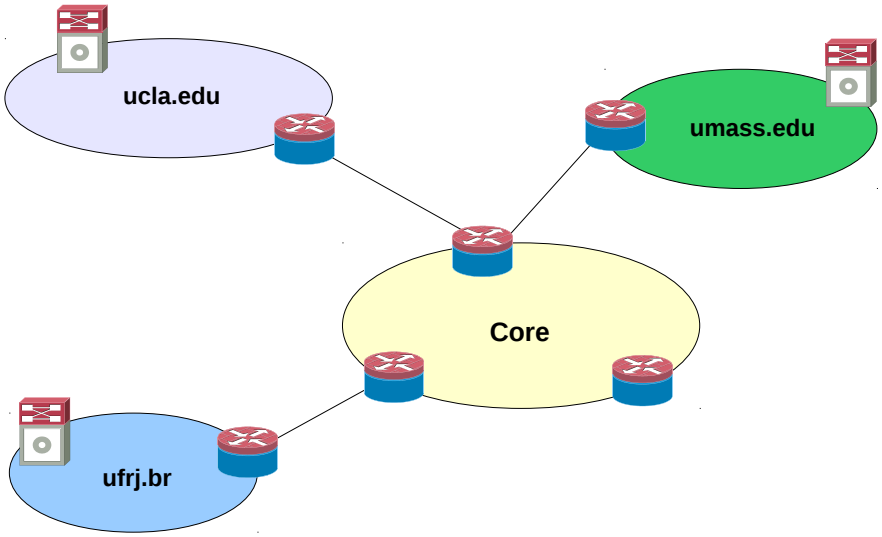


Figura 2.7: CCN Architecture

The FIB comprises name routing table mapping requests to an output interface(s). Requests are know as interests. The PIT stores pending interests and the CS operates as a cache for contents that have traversed the CR when sent to users. When an interest reaches a CR, the CR checks whether the prefix of the interest matches a content in its cache. In case a match is verified, content is sent to the requester. Otherwise, the CR checks if the interest is already stored in the PIT. If not, the interest is routed to a neighbor CR according to the routing table at the FIB. The check in the PIT before routing an interest avoids a CR to resend an interest already sent, but not yet served.

CCN is a coupled ICN approach: content requests and content data are routed by CRs. The CRs in a domain exchange prefixes through OSPF. A CR may end up with multiple interfaces in its FIB for the same prefix. BGP is proposed to announce prefixes across different domains. While requests are routed towards content storage locations they leave a trail along the CRs they have crossed to reach a content. Content follows the entire trail left by a request, when forwarded to a requester. In CCN, as content follows a trail, this trail is removed from the network. To enhance the discovery of cached contents, Rosensweig et al. [34] allows trails for previously downloaded contents to be preserved for some extra time, when content traverses the network.

Figure 2.8 shows contents being delivered from a custodian in ucla.edu to a host in ufrj.br, following the trail left by a request path, despite the same content being retrieved is stored in a closer AS. This is a result of forwarding request tables being composed by aggregated prefixes instead of full content names.

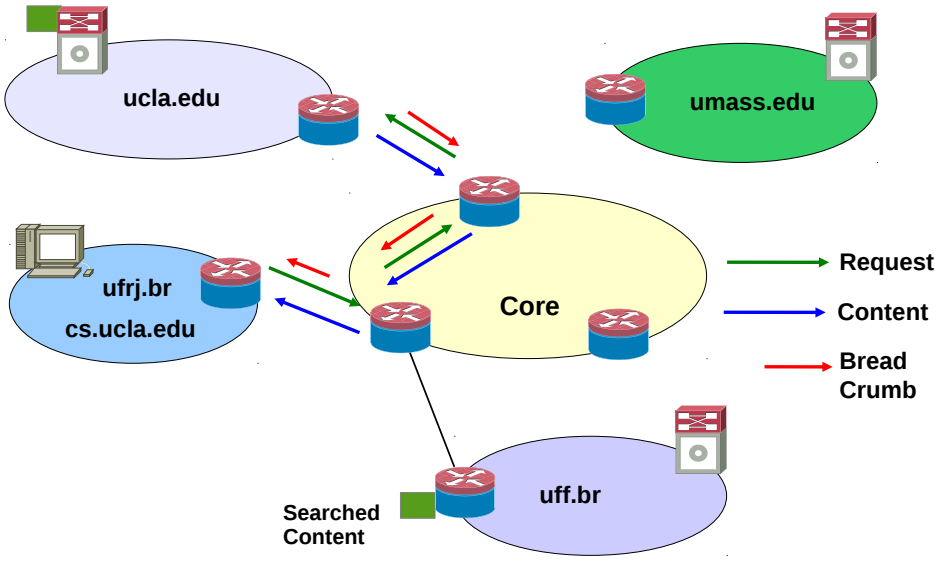


Figura 2.8: Request / Content Flow - CCN

2.4 ICN Open Issues

A set of open issues remains unsolved for ICNs [16]. In this work, we focus the challenges related to name resolution and content placement policies. Name resolution is responsible for translating content names to content storage locations. Content placement is responsible for establish how content is cached within network nodes so that the mean time to retrieve a searched content and the total load reaching custodians at publisher's networks is minimized. Our architectural proposal is described in chapter 3 and mathematical models in chapters 4 and 5.

A first strategy for name resolution considered so far in literature comprises a Pointer Resolution System (PRS). Whenever a PRS is deployed, requests are forwarded across a resolution network, upwards nodes storing pointer locators. At those nodes, a mapping between content names and content storage nodes is handled. After this mapping, requests are then forwarded to the content storage nodes. DONA and DHT architectures adopt PRS. In case PRS is adopted, how to set fast lookup policies for mapping names of data objects to content locations remains as an open challenge. In DONA, the top-level resolution servers are requested to retain the knowledge for all stored copies at the entire network. In DHT, this knowledge is distributed. Those architectures rely on topologies, structured according to a given geometry to route requests. The malfunctioning of a node may disrupt the entire structure behavior. In our proposal we do not rely on global knowledge at resolution servers for all stored copies at the entire network, neither geometry topologies to route requests.

A second strategy considered so far in literature comprises content routing tables (CRT). Whenever CRT are deployed, a direct path is established by routing tables towards content storage locations. CCN adopts CRT. Under CRT, the routing algorithm used for this approach heavily depends on the properties of the namespace. In case CRT is adopted, the main size of the content routing tables tends to get very huge. This size becomes a main concern. Name aggregation is a natural solution to be deployed, so that routing table sizes become feasible. Nevertheless, due to name aggregation, only name prefixes are maintained at the routing tables. If name aggregation is deployed, routing tables become unaware of cached replica identified by its full name.

For both PRS and CRT strategies, the main challenges remaining are: (i) How to fast update location of data objects expecting to change frequently, (ii) How information about replicas cached and evicted is broadcasted to the entire network, without imposing considerable communication costs. We do not rely on content routing tables to forward requests and contents, neither on pointer locators stored within the network to map requests issued to the network to content storage locations. Our proposal seeks to explore the vicinity of cache routers (exploration), the requests issued to the network reaches, as they flow to custodian networks wherein a copy of the desired content can be encountered (exploitation). Exploration versus exploitation trade-offs for ICN designs are discussed in [35]. Flooding is used as the exploratiom approach in [36]. In our proposal, we use random walks instead.

The caching approach used in the vast majority of existing ICN proposals is the Transparent En-Route Caching (TERC) [1] by which all ICN routers in the network participate in the process of content caching in conjunction with their primitive function of relaying the information objects downstream. This naive method of caching, however, has been subject of many controversies and criticisms [37], [38], [39]. To reduce caching redundancy, more complex varieties of this paradigm, such as probabilistic in network caching (ProbCache) [40]. We target opportunistic caching using reinforced counters, with a distinct time counter, for eviction purposes, for each cached content. Those counters and their dynamics are described in the next chapter.

Keeping track of up to date cached content at network level, is an open challenge as it heavily impacts scalability. Our strategy does not cope with broadcasting cache states to update a name resolution system. As caching takes place inside the network, different traffic compete for the same caching space. Our proposal presents more flexibility to adjust parameters for caching purposes, compared to others in the literature. We consider hysteresis, also described in the next chapter, for each reinforced counter. Hysteresis has not yet been considered in proposals dealing with a distinct time counter, for eviction purposes, for each cached content.

Capítulo 3

Proposed ICN Architecture

Our ICN architecture comprises a logical hierarchy consisting of tiers. Each router belongs to only one tier. Copies of popular contents may be cached into routers. We consider M logical hierarchical tiers, in which tier 1 is the top level tier, and tier M constitutes the bottom level. The top level tier of the hierarchy is the publishing area, which knows how to forward requests to at least one copy of the searched content. Each tiers is divided into a set of non-overlapping partitions called domains.

When a request arrives at a router in a tier, and the searched content is not found imediatly at this router, a search strategy, based on random walks (RW), is started within the tier domains the router that received the request belongs to. Random walks allow *opportunistic encounters* between requests and replicas in a best-effort manner and last for at most T time units in a domain. Requests, when flowing to the publishing area, leave a trail. Content flows along this trail in reverse direction, as it traverses the network towards users. This path is erased while the content is delivered.

We consider a special class of content placement mechanisms, named reinforced counters (RC). To each content we associate a RC in each router. The RC for a content is increased by one every time a request for this content reaches a router. The RC for a counter is decreased by one whenever a given timer ticks. Figure 3.1 displays routers forwarding requests towards a publishing area (green arrows). Opportunistic encounters prevent many requests to reach the publishing area. In case a searched content is not found after T elapsed, the request is randomly send to router in a

domain at the next higher tier of the hierarchy.

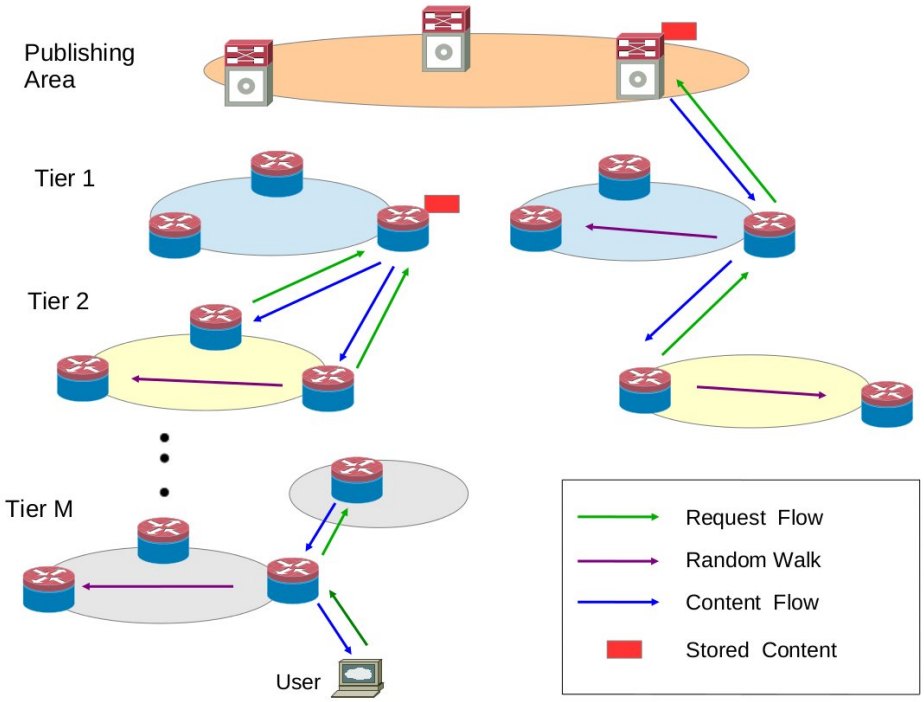


Figura 3.1: ICN proposed architecture

Our design is based on randomized algorithms for content search and caching along network nodes and was inspired by biological networks. Self-organized biological systems, where simple rules are performed by agents at individual level, through cooperation tasks, allow the emergence of complex pattern guiding complex tasks being performed [41], [42]. Biological systems present those properties as a result of an evolutionary process tailored for changing environment adaptive purposes.

We adopt randomized algorithms providing adaptive strategies, scalability and reliability [43], [44]. Network scalability imply capacity for adding resources without compromising network performance. The guidelines for network scalability comprise features such as: absence of global aware devices and global clock, decisions solely based upon local information exchange and failures of devices not disrupting the entire network. Uncentralized control mechanisms emerge as a consequence of local decisions based on information exchange under asynchronous communication fashion. Content replication with local caching offer extra support for uncentralized control mechanisms.

3.1 Random Walks

At the beginning of the twentieth century, random walks were described in the literature, under a discussion between Pearson and Rayleigh [45]. The first random walk models targeted movement description. In those walks, the actual direction of movement becomes completely independent of the movements in the past. This independence characterizes unbiased movements, described according to a Markovian process.

Unbiased walks stands for absence of preferred direction. At each step is, the direction taken by the walker is said to be completely random. Whenever the present movement takes place in any direction, this process describes a Brownian motion [46] producing standard diffusion. The unbiased random walk model is the basis of diffusive processes. Despite the simple rules guiding a walker in each step, random walk processes are relevant to understand many natural phenomena.

Many analitycal solutions to random walk processes requires the use of complex mathematical techniques [47]. Given a graph and a starting node, the random walk can be studied as if at each step a neighbor node is selected at random and the walker moves to this selected node. The random sequence of nodes a walk traverses establishes a path on the graph [48]. For random walks on graphs, there is not much difference between the theory of random walks and the theory of finite Markov chains. Quantitative aspects of the walk can be studied, similar to quantitative aspects across a one dimensional line:

- What is the distribution for the time a walker spends to hit a given node?
- What is the distribution for the time a walker spends to cross a given boundary?
- What is the distribution for the time a walker spends to cover the entire graph?

Walks with a consistent bias according to a preferred direction, towards a given target are named biased random walks. An example of biased random walks targeting enhancing searching performance on an unexplored area can be found in [49], while biased walks towards load balancing in [50]. Another example of biased random walks comprise ants randomly seeking to find food leaving pheromone trails as they wander around a colony (see Figure 3.2). An ant swarm represents

an example of self-organized biological system. In case food is discovered in a spot around the colony, the amount of pheromone between the colony and this spot is increased such that a direct path is build connecting both emerges as a new pattern. The pheromone left along trails connecting the nest and food is an example of biased random walks. Despite this direct path, some ants still continue exploration around the neighborhood of a colony, seeking for new food spots.

Figure 3.2 illustrates the direct path, with ants still exploring the neighborhood of the path to find new food spots. Ant colony optimization comprises a technique for solving several engineering problems [51], [52], [53], [54]. In our proposal, we deploy a biased walk, inspired by ACO. The requests flow according to a biased path towards a publishing area, but requests explore the vicinity of this path in tier domains, through random walks, in order to enhance the probability of opportunistic encounters aout of the biased path.

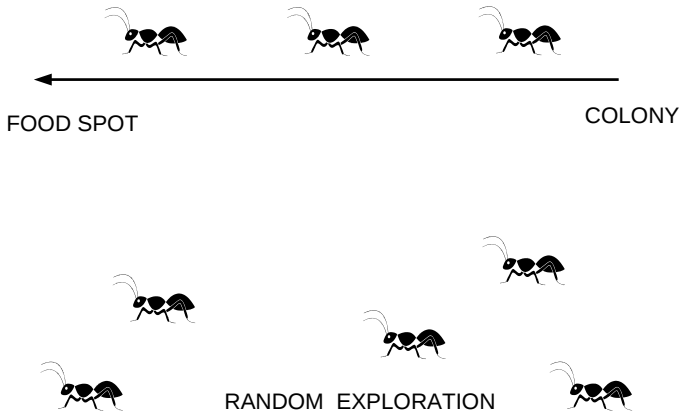


Figura 3.2: Ants

3.2 ICN: Probabilistic Search and Cache

Considering the number of content flowing in the Internet nowadays, caching/eviction policies at network level impose relevant challenges in terms of scalable and resilient strategies. Pre-defined knowledge at name resolution systems about the replicas at caches in ICNs has been considered through the adoption either of pointer locators mapping names to storage locations or routing tables establishing direct paths to storage locations. If content routing tables are considered, searched content can reach caches within the path between the requester and a custodian for the content. In this case, nearby replicas stored at the vicinity of the path may not be reached.

The debate in [35] has opened the door to ICN designs to enhance cache hits considering sending part of the requests over direct paths, to known stored copies (exploitation), while part of the requests are forwarded through random walks to seek for opportunistic encounters (exploration). While exploitation is guaranteed to route a request to a known copy, exploration does not solely guarantees a searching success. Also, the rate of success of exploration depends on how long the random walk may last and the replica density placement at storage areas being explored by the walker.

The debate in [40] has opened the doors for probabilistic algorithms targeting content placement across network nodes. Caching capacity spread along the network nodes is small compared to the total number of different contents flowing in ICNs. Avoiding cache redundancy becomes a key policy design. With probabilistic strategies favoring a better content utilization of storage resources, in many scenarios, more space left was achieved for different flows sharing same storage resources.

Figure 3.3 shows a direct path to a content being exploited, while some exploration around the vicinity of cache C2 is issued. The direct path traverses a set of cache routers in a cloud. Therefore, a trade-off is established. For popular contents, the highest the exploration time, the highest the number of opportunistic encounters. Nonetheless, the mean time to find a content gets incremented. The direct path presents a much higher cost for requests to reach the searched content, compared to requests reaching content in cache C3. The exploration is held in unknown areas to the routing tables, what may lead to a unsuccess. When content is delivered to

users, probabilistic decisions may take place to decide whether this content should be stored in a cache traversed by the contents.

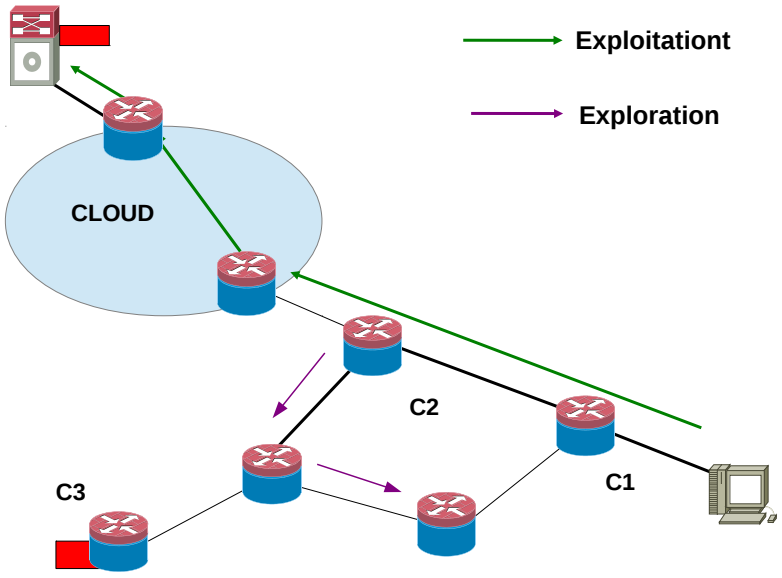


Figura 3.3: Exploitation versus Exploration

3.3 Thesis ICN Design

Our search strategy deploys random walks (RW) across a single domain in a tier. When a request arrives in a domain and does not find imediatly the searched content, the router that has received the request holds the request and issue a random walk, traversing only routers in the same domain. Routers within a domain are assumed to be logically connected. The walk lasts for at most T units of time in a domain. In case a walk fails in a domain, the request is send randomly to a cache in a domain at the next level of the hierarchy. The last tier is the publishing area. (see Figure 3.1).

The goal is to opportunistically explore the presence of content replicas in a given domain. If a copy is found in the domain within a reasonable time interval, the content is served. Otherwise, requests are routed, from one tier domain to another tier domain following the hierarchy of the architecture. At the top level of

the hierarchy a publishing area knows how to forward a request, not able to get a cache hit, to at least one copy of the desired searched content.

When the requests are routed to the publishing area, they leave a trail. The first router at each different domain a request reach will compose such trail, named bread crumbs. When content is located in the network, it is delivered back to the users following the reverse path of the bread crumbs. As the content follows the path of *bread crumbs*, the trail is erased.

In the network, each cache-router has a given service capacity. The service capacity of a cache is related to the time it takes to 1) find content in the cache, 2) return that content to the user, through the data plane, in case of a cache hit and 3) route the request to another cache, through the control plane, in case of a cache miss.

Network of caches with LRU, FIFO and RND policies are very difficult to analyze. A new class of caches named time to live caches (TTL-caches) have been proposed to this aim [55]. It has been shown that decoupling the dynamics of multiple contents can lead to reasonable approximations to the content hit probabilities. For TTL-caches, an eviction timer is associated to each content. After a content has been stored, this content is evicted as soon as this timer has elapsed. They are also of interest in the context of DNS caches and the novel Amazon ElastiCache system (`http://aws.amazon.com/elasticache/`).

To efficiently and distributedly place content in the cache network, we consider a flexible content placement mechanism inspired by TTL-caches. At each cache, to each content there is an associated counter, which we call reinforced counter (RC). The value of a counter indicates if the related content should be stored into cache provided that enough storage is available. The reinforced counters are increased by one every time the content is requested, and is decreased by one as a timer ticks. The timer ticks every $1/\mu$ seconds. Henceforth, we assume that the time between ticks is exponentially distributed. When cache capacity is finite, one can take advantage of statistical multiplexing either by evicting items from the cache when (a) an overflow occurs or (b) when the item expire.

Each published content in the network is identified by a unique hash key inf. All cache routers have a set of RCs, each RC for a different content. The RCs have two

thresholds: an upper K_{max} and a lower K_{min}. Whenever a request for the content *inf* reaches a router, either (i) an already pre-allocated counter for *inf* is incremented in this router by one or (ii) a free counter is allocated for *inf* in this router and incremented by one. Whenever the RC for *inf* reaches K_{max}, if the content for *inf* is not cached at this router, this content gets cached when delivered to users. In contrast, whenever the RC for *inf* reaches K_{min}, if the content for *inf* is cached at this router, this content gets immediately removed and the correspondent counter is deallocated. Figure 3.4 depicts the dynamics of the RC counter.

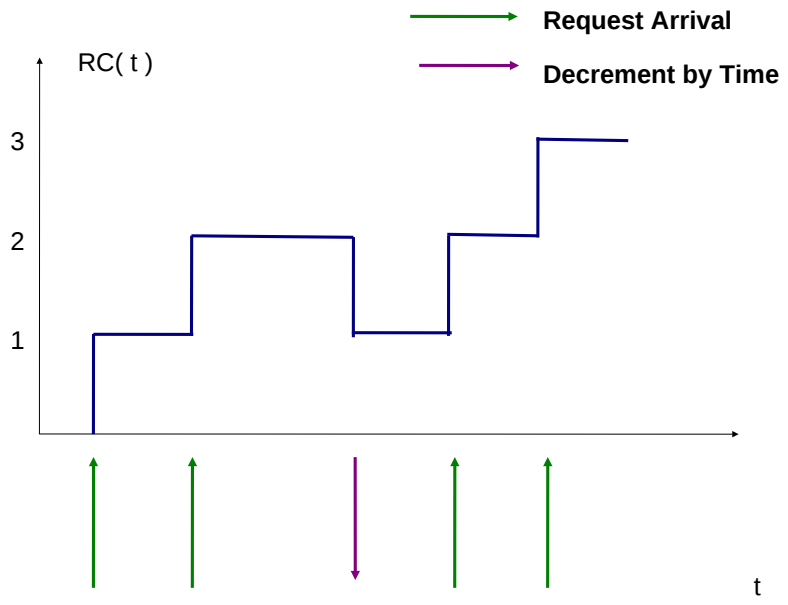


Figura 3.4: RC dynamics

We decouple the dynamics for different contents. RCs dynamics are independent of each other. Therefore, the mathematical analysis for a single content *inf* turns to be the same for all contents. Instead of considering that the cache has a fixed capacity, it is assumed that there are constraints on the expected number of items in the cache. Note that under this approximation the cache capacity design problem can be compared to the problem of calculating the capacity of communication lines in a telephone network. Roughly, when capacity is finite, one can take advantage of statistical multiplexing either by evicting items from the cache when an overflow occurs or when the item expire.

We aim to show the advantages of having a content placement mechanism with

control knobs entangled by the RC thresholds and the decreasing RC rate (RC parameters), allowing a fine tune for both the fraction of time in which the content is in the cache and the the rate at which content is evicted or brought back into cache, so as to avoid that content is replaced too fast and content starvation (that is, content is never included into cache or removed from it during a finite but large time interval). Also we target to minimize both the cost for content to be transferred from custodians to caches and the cost for a user to retrieve a searched content from within the network and maximize the hitting rate at the caches so to decrease the total load entering the publishing area.

Given such architecture, we pose the following questions:

1. while routing requests for content from users to custodians, how to determine for how long a request should search for a content inside a domain in order to optimize the performance metrics?

2. while placing contents at a cache-router, how to tune the reinforced counters parameter values to optimize the delay to retrieve content under storage constraints and possibly other performance metrics of interest?

3. what are the parameters that have the greatest impact on performance metrics of the ICN architecture considered?

To answer these questions, we propose an analytical model which yields: a) the expected delay to find a content (average search time) and; b) the rate at which requests have to be satisfied by the custodians. While the first metric is directly related to users quality of experience, the latter is associated to publishing costs by custodians. The model yields simple closed-form expressions for the metrics of interest. In addition, we address the content eviction rate from cache which may have an adverse impact on performance. Using the model, we study different tradeoffs involved in the choice of parameters values. In particular, we study the tradeoff between spending more or less time in exploring opportunistically around the user vicinity in order to find content replicas. Optimal values for the performance metrics can be obtained whenever constraints such as maximum buffer size and maximum cache writing rates are set, as depicted in the following chapters.

Capítulo 4

Local Performance

In this chapter we study placement and eviction policies for a single cache under the assumption that the dynamics of each content is decoupled from the others. Decoupled content dynamics yields tractable analysis and can be used to approximate the performance of systems with fixed capacity. We consider two polices: one where the reinforced counter has the same threshold to insert and remove content from cache, and other where the reinforced counter has two thresholds, one to insert and other to remove content from cache which we call reinforced counter with hysteresis.

Using the model defined for the dynamics of the reinforced counter, we show the advantages of having a content placement mechanism with control knobs to fine tune both the fraction of time in which the content is in the cache while at the same time controlling the mean time between content insertions so as to avoid that content is replaced too fast and content starvation. The control knobs can be tuned so as to adjust the long term fraction of time in which the content is stored in the cache and to guarantee that the mean time between content eviction and content reinsertion into the cache is bounded.

The reinforced counter (RC) dynamics of different contents can be assumed uncoupled. In addition, arrival requests for a content at different routers are assumed independent arrival processes and the search for a content does not interfere with the dynamics of the corresponding RC counter. Consequently, the caches in a domain can be treated independently.

Cache hit ratio is a key metric to performance. However, another metric that affects performance is the rate at which content is brought into cache from the

server (cache insertion rate). A smaller insertion rate, for the same hit ratio, has several advantages: (a) first, increasing the number of cache writes slows down servicing the requests for other contents, that is, cache churn increases which reduces throughput [56], [57]; (b) if flash memory is used for the cache, write operations are much slower than reads and; (c) writes wear-out the flash memory; (d) additional writes mean increasing power consumption. Another advantage may also occur if content delivery to user and cache insertion are done in parallel, using different paths, since this will increase the network load.

4.1 Reinforced counter mechanism with a single threshold K

For a given content, we model the dynamics of the reinforced counter as a birth-death process at each cache. The birth rate is set by the arrival rate λ of requests to the cache and the death rate by an exponential random variable with rate μ, a parameter that can be set by the system tailored for best performance.

Figure 4.1 is useful to illustrate the different intervals of the cache content replacement and the notation we use. The blue (red) intervals in the figure indicate that the content is stored (or not) in cache. If content is not in cache it is brought into cache when a new request for it arrives and the reinforced counter is at the threshold K. On the other hand, if the value of the reinforced counter is at $K+1$ and the counter ticks, the content is removed from cache. Note that we adopt the same assumption as in [55]: the insertion and eviction of a content is not influenced by other contents in the same cache.

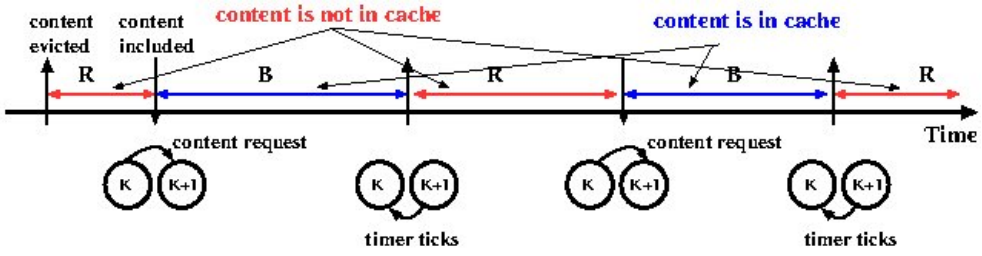


Figura 4.1: Request counter and notation.

It should be clear that the reinforced counter mechanism can be modeled as an

M/M/1 queueing system, with state equal to the value of the reinforced counter. We define $\pi_{up}(K)$ as the fraction of time the content is in the cache given a threshold K. Then,

$$\pi_{up}(K) = \sum_{i=K+1}^{\infty} (1-\rho)\rho^i = \rho^{K+1} \tag{4.1}$$

where $\rho = \lambda/\mu$.

Each content alternates from periods of inclusion and exclusion from the cache according to the value of the threshold K of the reinforced counter (see Figure 4.1). $\pi_{up}(K)$ corresponds to the fraction of time in which a content is in the cache. Then $\pi_{up}(K)$ can be computed from (renewal theory):

$$\pi_{up}(K) = \frac{E[B]}{E[R] + E[B]} \tag{4.2}$$

where $E[R]$ is the mean time that a content takes to return to the cache once it is evicted and $E[B]$ is the mean time that the content remains in the cache after insertion. Let $\eta(K, \mu)$ be the rate at which content enters the cache. Due to flow balance, in steady state, $\eta(K, \mu)$ equals the rate at which content leaves the cache,

$$\eta(K,\mu) = \mu\rho^{K+1}(1-\rho) = \lambda\rho^K(1-\rho) = \frac{1}{E[B] + E[R]} \tag{4.3}$$

where the last equality can be easily inferred from Figure 4.1 (renewal arguments).

$E[B]$ can be calculated from first passage time arguments, that is the time it takes from the system to return to state K (eviction) once it is brought into system (state $(K+1)$). From the M/M/1 model, it can also be calculated by the busy period of an M/G/1 queue, which equals

$$E[B] = 1/(\mu - \lambda). \tag{4.4}$$

Then, from (4.4) and (4.2)

$$\pi_{up}(K) = \frac{1/(\mu-\lambda)}{1/(\mu-\lambda) + E[R]}. \tag{4.5}$$

Once the content is evicted the mean time for it to return to the cache is given by

$$E[R] = (1 - \pi_{up}(K))/(\pi_{up}(K)(\mu - \lambda)) \tag{4.6}$$

Given a fixed $\pi_{up}(K)$, it is possible to write ρ, μ, $E[R]$ and η as a function of K,

$$\begin{aligned} \rho &= \pi_{up}(K)^{1/(K+1)} && (4.7)\\ \mu &= \lambda\pi_{up}(K)^{-1/(K+1)} && (4.8)\\ E[R] &= (1-\pi_{up}(K))/(\pi_{up}(K)(\lambda(\pi_{up}(K))^{-1/(K+1)}-\lambda)) && (4.9)\\ \eta &= \lambda\pi_{up}(K)((\pi_{up}(K))^{-1/(K+1)}-1) && (4.10) \end{aligned}$$

Note that there is a tradeoff in the choice of K, as increasing the value of K reduces the rate at which content is inserted into the cache (see equation (4.3)), which in turn reduces the steady state costs to download the content from external sources. The larger the value of K, the smaller the steady state rate at which the content enters and leaves the cache. But increasing the value of K, also increases the mean time for the content to be reinserted into the cache once the content is evicted. The larger the value of K, the longer the requesters for a given content will have to wait in order to be able to download the content from the cache after it is evicted (see equation (4.9)).

Let K_{max} be the maximum value allowed for K. Motivated by the tradeoff above, given a fixed value of $\pi_{up}(K)$ we consider the following optimization problem,

$$\begin{aligned} \min_{K} \quad & \psi(K) = \alpha\eta + \beta E[R] && (4.11)\\ \text{such that} \quad & && (4.12)\\ K &\leq K_{max} && (4.13)\\ \pi_{up}(K) &= \rho^{K+1} && (4.14)\\ \rho &= \lambda/(\lambda\pi_{up}(K)^{-1/(K+1)}) && (4.15) \end{aligned}$$

α and β are used to control the relevance of long term and short term dynamics. The long term dynamics reflect the behavior of the system after a long period of time, during which the rate at which content enters the cache is given by $\mu\rho^{K+1}(1-\rho)$. The short term dynamics reflect the behavior of the system during a shorter period of time, during which one wants to guarantee that the mean time it takes for the content to return to the cache is not too large. We should keep in mind that we choose $\pi_{up}(K)$ to satisfy cache capacity limitations and system performance.

The value of $E[R]$ must be bounded so as to avoid starvation, as formalized in the proposition below.

Proposition 4.1. *If $\beta = 0$, the optimal strategy consists of setting $K = K_{max}$. If $K_{max} = \infty$, it will take infinite time for the content to be reinserted in the cache once it is evicted for the first time.*

Proof: The objective function is given by

$$\psi(K) = \alpha\lambda\pi_{up}(K)((\pi_{up}(K))^{-1/(K+1)} - 1) \tag{4.16}$$

The derivative of the expression above with respect to K is

$$\frac{d\psi(K)}{dK} = \alpha\frac{\lambda\log(\pi_{up}(K))\pi_{up}(K)}{(K+1)^2(\pi_{up}(K))^{1/(K+1)}} \tag{4.17}$$

which is readily verified to be always negative. Therefore, the minimum is reached when $K = K_{max}$. When $K = \infty$ it follows from (4.8) that μ tends to 0. The mean time for reinsertion of the content in the cache is given by (4.9) which grows unboundedly as μ tends to 0. □

We consider an illustrative example to show the tradeoff in the choice of K between the rate at which the counter reaches the upper threshold and the mean time a content is cached again after evicted. Let $\pi_{up} = 0.9$ and $\alpha = \beta = 1$. Figure 4.2 shows how cost first decreases and then increases, as K increases. The optimal is reached for $K = 10$. At that point, we have $E[R] = 0.31$ and $\gamma = 0.32$.

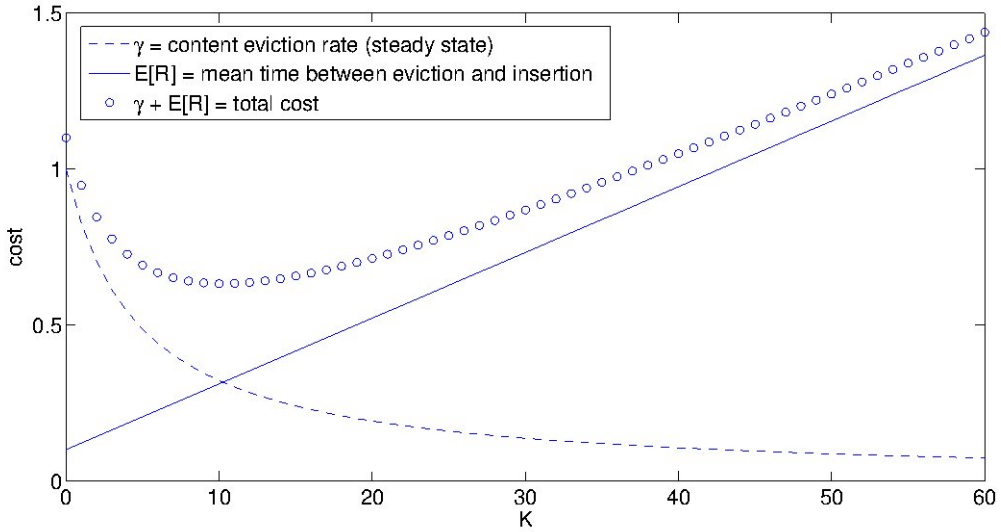


Figura 4.2: $\pi_{up} = 0.9$, $\alpha = \beta = 1$

4.2 Reinforced counter with hysteresis

In this section we generalize the reinforced counter to allow for a third control knob K_h as follows. The counter is incremented at each arrival request for a content and is decremented at rate $1/\mu$. In addition, the content is included into cache when the value of the reinforced counter is incremented to $K+1$, as in previous section. However, content is not removed from cache when the counter value is decremented from $K+1$ to K. Instead, content remains in cache until the counter reaches the (new) threshold K_h. Figure 4.3 illustrates the behavior of the new reinforced counter.

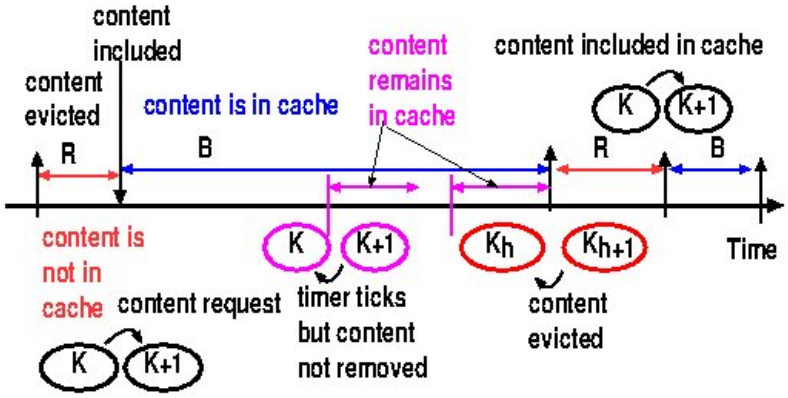


Figura 4.3: Reinforced counter with hysteresis.

From the figure, we observe that the purple intervals correspond to the values of the reinforced counter between K and K_h and the content is in cache. The bottom of Figure 4.3 shows a trajectory of the reinforced counter that motivates the name *hysteresis*. The displayed trajectory is $i \to K_h \to K_h+1 \to K \to K+1 \to j \to K_h+1 \to l$ etc.

At first glance, it seems that only the intervals where the content is in cache are affected by this new mechanism (the blue intervals) but not the intervals where content is absent (the red intervals). However, this is not true and both intervals are affected. In what follows, we show how to calculate the expected values of $E[B]$ and $E[R]$ and how the measures of interest are affected by this new mechanism. We also show the advantages of the reinforced counter with hysteresis. For that, we refer to Figure 4.4 that shows the Markov chain for the hysteresis counter.

Let $\nu(i+1) = E[B]$ and $\xi(i+1) = E[R]$ both when $K - K_h = i$. Note that

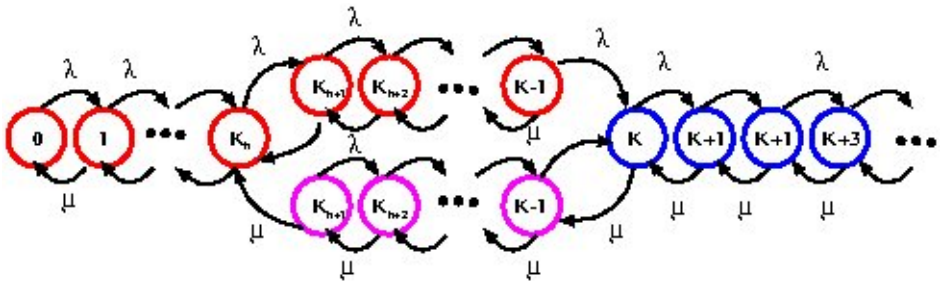


Figura 4.4: Reinforced counter with hysteresis: state transition diagram.

$\nu(1)$ and $\xi(1)$ are the values of $E[B]$ and $E[R]$ for a cache with no hysteresis. $\nu(1)$ is obtained from equations (4.2), (4.1) and (4.9). $\xi(1) = \left(\sum_{j=0}^{K} 1/\rho^j\right) / (\lambda/(\lambda+\mu))$.

Proposition 4.2. *$\nu(i+1)$ and $\xi(i+1)$ can be obtained by the following recursions:*

$$\nu(i) = \tfrac{1}{\mu} + \nu(i-1) + \rho\nu(1) \qquad i \geq 2 \tag{4.18}$$

$$\xi(i) = \tfrac{1}{\lambda} + \xi(i-1) + \tfrac{1}{\rho}\xi(1) \qquad 2 \leq i \leq K - K_h + 1 \tag{4.19}$$

Proof: The proof follows from renewal arguments. □

It is important to note that explicit expressions for $\nu(i)$ and $\xi(i)$ can be obtained as a function of λ, μ and K but details are omitted since the recursion above suffices to explain our comments.

Suppose λ and $\pi_{up}(K)$ are given and we obtain K and μ, for instance from the optimization problem in the previous section. We allow K_h to vary from $K_h = K$ (that is reinforced counter without hysteresis) to $K_h = 0$.

Proposition 4.3. *As K_h decreases, the rate $\eta(K,\mu)$ at which content enters the cache also decreases.*

Proof: From Proposition 4.2, it is not difficult to see that both $E[B]$ and $E[R]$ increase with K_h. Since $\eta(K,\mu) = 1/(E[B] + E[R])$ (equation (4.3)) the result follows. □

Proposition 4.3 shows that, from an initial value of $\pi_{up}(K)$, if we fix the parameters λ and K, the rate at which content is replaced (both included and removed from cache) decreases by using the hysteresis mechanism, which is good to lower costs as

explained in the previous section. However, $\pi_{up}(K)$ also varies. As a consequence of Proposition 4.2 we can show that and $\pi_{up}(K)$ can be reduced. This is not obvious since $E[B]$ increases. But, by adjusting the knob μ, $\pi_{up}(K)$ can be maintained constant while η is reduced when the hysteresis schema is used. The proof of this last result is omitted but it follows from Proposition 4.2.

There are additional advantages of the hysteresis mechanism. First note that, in the previous sections, we assumed that file download times are negligible. However, when a user requests for a content it is important that the whole file remains stored at the cache not only until this user finishes downloading but also while other users are downloading the same content from that cache. Hysteresis is helpful to prevent the file from being removed before its download is concluded by all requesters. Hysteresis increases the probability that a content remains in cache for at least some time t after it is cached. Note that this is an additional (transient) performance measure and it differs from the η metric (steady state rate). As our numerical results show, by adjusting K_h we can improve both steady state and transient metrics.

Next, we show some numerical results obtained for the following scenarios: (a) the reinforced counter has a single threshold $K = 11$, (b) the reinforced counter has two thresholds $K = 11$ and $K_h = 7$, and (c) the reinforced counter has two thresholds $K = 11$ and $K_h = 2$. For each scenario, we consider the same value of π_{up}, λ, and K. The value of γ for scenario (a) is 0.24, for (b) is 0.07 and for (c) is 0.04. We note that the rate at which content enters or leaves the cache decreases as the value of K_h decreases. This is one of the advantages of introducing a third control knob K_h.

Figures 4.5 and 4.6 show the cumulative distribution of R and B. We note that the reinforced counter with hysteresis allow to control the probability distribution of R and B. In Figure 4.5, consider for example $t = 4$. If we set $K_h = 2$, we have $P[R < 4] = 0.35$ and if $K_h = 7$, then $P[R < 4] = 0.65$. As the value of K_h increases, the probability of R be less than a certain value of t increases. This behavior can also be observed for the distribution of B. On the other hand, if we consider the model with a single threshold we can not control the distribution of R and B. In Figure 4.5, $P[R < 4] = 0.9$.

Another advantage of the reinforced counter with hysteresis, is that the coefficient

of variation of R and B, decreases with the value of K_h, which means that the dispersion of the distribution of R and B also decreases. The values obtained for the coefficient of variation of B for each scenario are: (a) 1.6, (b) 1.3 and (c) 1.1.

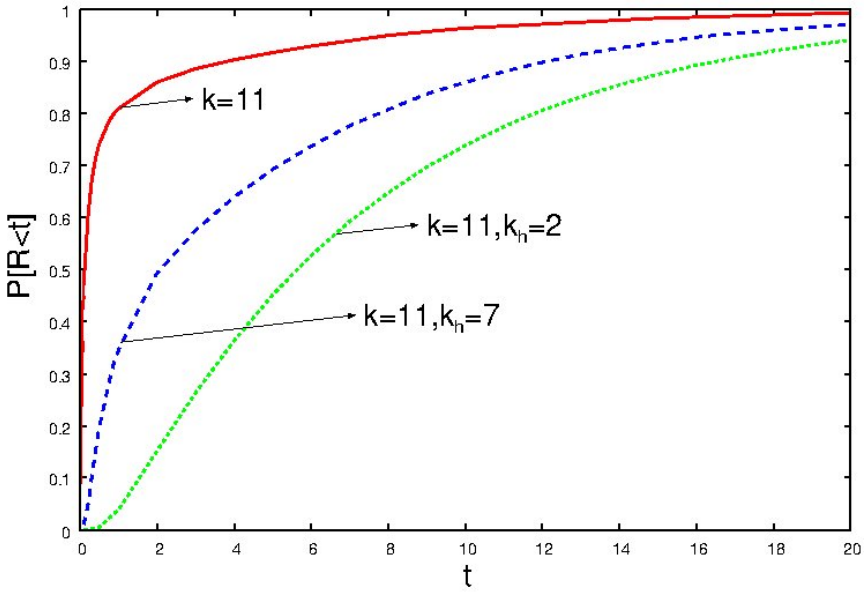


Figura 4.5: Cumulative Distribution of the time the content takes to return to the cache

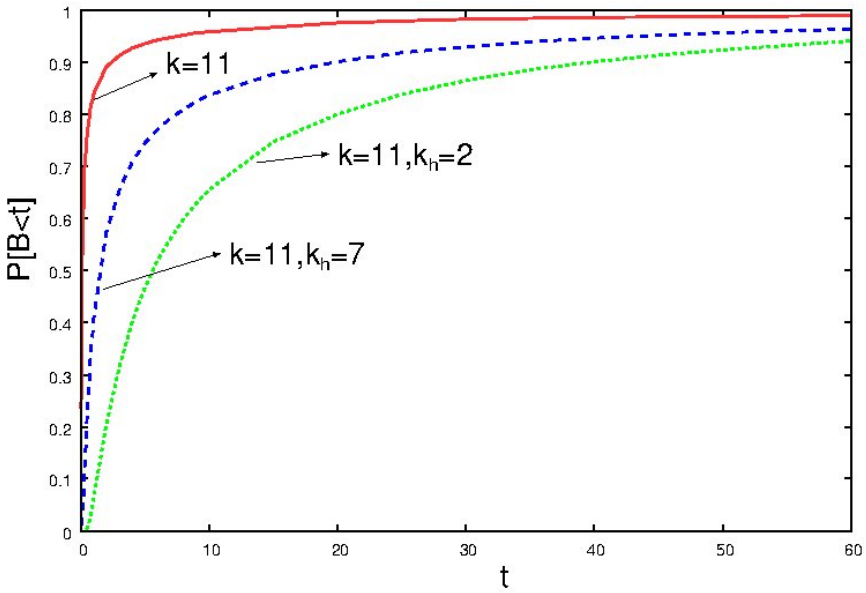


Figura 4.6: Cumulative Distribution of the time the content takes to remain in cache

Capítulo 5

Network Performance

We propose analytical models that take into account both the performance impact of the delay introduced when content is searched using random walks and the caching mechanism based on the reinforced counters. We seek to obtain the following performance goals: (i) to reduce the total load reaching the publishing area, (ii) to enhance the mean time to find popular content, and (iii) to reduce the rate at which content enters and leaves the cache, which in turn reduces the cost to download content from the publishing area.

We develop analytical models which yield closed form expressions for performance metrics across the entire network. The analytical models are inspired by reliability theory concepts [58]. Metrics such as the mean time to find a content and the probability of not finding a content in a domain can be computed from the models. The models allow to show the benefits of a logical hierarchy of tiers and study the tradeoff between the storage requirements of a cache-router and the total load that reaches the storage area.

When a request reaches a cache-router, the local cache is searched and the content is immediately retrieved and sent to the user if it is locally stored. If the content is not found, a random walk search starts in the domain. We assume that the random search takes V time units per each cache-router visited where V is a random variable exponentially distributed with rate γ. (Clearly, the smaller is the probability of finding a content in a cache-router, the longer is the duration of the search in order to find the content.)

Since longer search times have an adverse effect on performance, when the ran-

dom walk begins, a timer is set to limit the search time. At most the search can last for T time units. The search finishes when the timer expires or the content is found, whichever occurs first. When the timer expires, the user request is sent to the next cache-router in the tier hierarchy and the process starts over.

Consider a given tagged cache-router. We assume that requests to content c arrive to the cache-router according to a Poisson process with rate λ_c. λ_c is also referred to as the content popularity. The Poisson assumption is not unrealistic for our purposes. This assumption has been used in several modeling works.

In addition, recent work in [59] using three months of data from the largest VoD provider in Brazil, indicates that, during peak hours, the Poisson distribution models fits quite well the request rate for all the movies (one minute beams were considered). We use the Poisson assumption coupled with a proper popularity model for individual content, such as the Zipf distribution.

Below we adopt the same assumption as in [55]: the insertion and eviction of a content is not influenced by other contents in the same cache. Fixed storage in cache is modeled by considering the expected number of contents into the cache.

We consider a domain wherein N cache-routers are logically fully connected, i.e., any cache-router can exchange messages with any other router in the same domain. Our goal is to compute the probability $R(t)$ that a random walk does not find the requested content by time t, which causes the request to be forwarded to the custodian.

We consider slightly different models. The models assume that a request for a content arrives according to a Poisson process. In what follows we describe the assumptions used for each individual model, and comment on their usefulness.

Table 5.1 summarizes the notation used in the remainder of this paper.

Parameter	Description
$1/\gamma$	average time for the random search to check for a content at a cache
$\mathcal{C}(\hat{\Lambda}_c)$	cost function incurred by custodian to serve a request (measured in delay experienced by users)
C	number of contents
M	number of tiers
N	number of caches in domain under consideration

$\lambda_{c,i}$	arrival rate of exogenous and interdomain requests for content c at typical cache of domain i, $\lambda = \sum_{i=1}^{M} \sum_{c=1}^{C} \lambda_{c,i}$
Λ_c	exogenous arrival rate of requests for c at the network (except otherwise noted, exogenous requests are issued at tier M)

Variable	Description
L	number of replicas of given content in tagged tier
$\pi_{c,i}$	probability that content c is stored at typical cache at domain i
$\alpha_{c,i}$	$= 1/\mu_{c,i}$

Control variable	Description
$\mu_{c,i}$	reinforced counter decrement rate for content c at domain i
$T_{c,i}$	maximum time to perform a random search for content c at domain i

Metric	Description
$R_{c,i}(t)$	probability of not finding content c at tier i by time t
$D_{c,i}$	delay incurred for finding content c at tier i
D_c	delay incurred for finding content c
D	delay incurred for finding typical content
$\hat{\Lambda}_c$	rate of requests for content c at the publisher

Tabela 5.1: Table of notation.

5.1 Model 1: Stateless Search

We call *stateless search* the content search that doe not carry any information on previous visited cache-routers. In other words, when a router is visited, the only information that is known is the content of the cache current being visited. All information on the contents of the cache routers previously visited is not stored. The first model considers this case.

We assume that the search that starts at router i is sufficiently fast so that the probability that there is a content c change in any cache-router of that domain is negligible. This assumption is reasonable if the elapse time $(1/\gamma)$ it takes for the random walker to move from a cache-router to another and to check for c is very small compared to the time interval it takes between: (a) two requests for c $(1/\lambda_c)$ and; (b) decrements of the reinforced counter for c $(1/\mu_c)$.

A request for content c that arrives at cache-router i (for any cache-router i

in the domain) sees the system in equilibrium (PASTA property) because of our previous assumption that the request arrival process is Poisson. We further assume that the rate of requests for a given content at different routers in a domain are approximately identical. When a request for a given content c arrives at a cache-router and a miss occurs, a random stateless search for c starts, looking for c in the remaining $N-1$ routers in the domain. At each router visited the searcher selects at random one of the remaining $N-1$ routers to visit. Note that, because the search is stateless, nodes can be revisited during the random search.

In what follows we drop the dependence of all variables to c to simplify notation. Let π be the probability that a given cache router stores the content of interest. In Chapter 4 we argue that the reinforced counter (RC) dynamics of different contents can be assumed uncoupled and so the π's for each content are independent. In addition, arrival requests for a content at different routers are assumed independent arrival processes and the search for a content does not interfere with the dynamics of the corresponding RC counter. Consequently, the caches in a domain can be treated independently.

Let L be random variable equal to the number of replicas of the content c in the domain, excluding the router being visited. We have:

$$P(L=l) = \binom{N-1}{l} \pi^l (1-\pi)^{N-1-l}. \tag{5.1}$$

Let J be the number of hops traversed by the stateless searcher by time t. Since the time between visits is assumed to be exponential distributed,

$$R(t|J=j, L=l) \;\; = \;\; (1-\pi)(1-w_l)^j \tag{5.2}$$

where w_l is the conditional probability that the random walker selects one router with content from the remaining $N-1$ routers in the domain, given that there are l replicas in the domain and the visited router does not have the content. Then, $w_l = l/(N-1)$. (Note that π depends on the placement policy and its parameter values.)

Proposition 5.1. *The probability $R(t|L=l)$ is given by:*

$$R(t|L=l) \;\; = \;\; (1-\pi)e^{-\gamma\omega_l t} \tag{5.3}$$

Proof: From (5.2) we obtain:

$$\begin{aligned} R(t|L=l) &= (1-\pi)\sum_{n=0}^{\infty}\frac{(\gamma t)^n}{n!}(1-\omega_l)^n e^{-\gamma t} \\ &= \frac{1-\pi}{e^{\gamma t\omega_l}}\sum_{n=0}^{\infty}\frac{(\gamma t(1-\omega_l))^n}{n!}e^{-\gamma t(1-\omega_l)} \\ &= (1-\pi)e^{-\gamma\omega_l t} \end{aligned} \tag{5.4}$$

□

Proposition 5.2 (Stateless search)**.** *The probability $R(t)$ a walker does not find a requested tagged content in a domain by time t is given by:*

$$R(t) = \left(e^{-\gamma t/(N-1)}\pi + (1-\pi)\right)^{(N-1)}(1-\pi) \tag{5.5}$$

Proof: Unconditioning (5.3) on L, we obtain:

$$\begin{aligned} R(t) &= \sum_{l=0}^{N-1} R(t|L=l)\binom{N-1}{l}\pi^l(1-\pi)^{(N-1-l)} \\ &= (1-\pi)\sum_{l=0}^{N-1} e^{-\gamma\omega_l t}\binom{N-1}{l}\pi^l(1-\pi)^{(N-1-l)} \\ &= (1-\pi)\sum_{l=0}^{N-1}\binom{N-1}{l}\left(e^{-\gamma t/(N-1)}\pi\right)^l(1-\pi)^{N-1-l} \\ &= (1-\pi)\left(\pi e^{-\gamma t/(N-1)} + (1-\pi)\right)^{(N-1)} \end{aligned} \tag{5.6}$$

□

According to (5.5), $R(\infty) = (1-\pi)^N$. As the random walk time increases, the probability that the walker does not find the searched content approaches the probability that all N caches within the domain do not hold the searched content.

5.2 Model 2: Statefull search

We call *statefull search* that in which the random searcher knows routers that are visited. In other words, no re-visiting is allowed, since we know that previous routers visited where found with no content in them. Two possible ways to implement such statefull search are: (a) when a request arrives at a router and finds that a request cannot be satisfied, a search is initiated and the searcher pre-selects j out of the remaining $N-1$ routers to conduct the search and; (b) after the search is initiated,

the searcher chooses the next router to visit at random, from those that have not yet been visited before.

We first consider the case in which routers are pre-selected at the beginning of the search. Let J be the number of routers that can possibly be visited by time t, and L as before. Conditioning on J pre-selected routers and l contents in the $N-1$ possible caches to visit,

$$R(t|J=j, L=l) = (1-\pi)\frac{\binom{N-1-l}{j}}{\binom{N-1}{j}} \tag{5.7}$$

Assuming, like before, that the search is sufficiently fast compared to the rate of changes in the RC counters and using (5.1),

$$\begin{aligned} R(t|J=j) &= \sum_{l=0}^{N-1} R(t|J=j, L=l)\binom{N-1}{l}\pi^l(1-\pi)^{N-1-l} \\ &= (1-\pi)\sum_{l=0}^{N-1}\binom{N-1-j}{l}\pi^l(1-\pi)^{N-1-l} \\ &= (1-\pi)\sum_{l=0}^{N-1-j}\binom{N-1-j}{l}\pi^l(1-\pi)^{N-1-l} \\ &= (1-\pi)^{j+1} \end{aligned} \tag{5.8}$$

Note that, if we select $J > N-1-l$ routers, necessarily one of them will have the content. Therefore the third equality is true since, when $l > N-1-j$, $R(t|J=j) = 0$.

It remains to uncondition on J, but we defer this step to later in this section. First we address the other possible model of the statefull search. As in the first stateless model, that no cache-router is revisited by a random walk search. However, we do not assume that the rate γ is very large compared to the rate of changes in the RC counters.

As before, a request for content c that arrives at cache-router i triggers a random search if the content is not found at i. In other words, a search for c at i is initiated according to the state of the RC counter for c at i, that is when the RC counter value is below the threshold established for c. As in previous models, the number of requests for content c that arrive at a router is assumed to be a Poisson distributed random variable. Therefore, the number of random searches for c that are initiated at router i is a Poisson process modulated by the RC counter values. (Recall that

the RC counter is a markovian process.) Since the RC counters for c at different routers are not affected by the random search, they are independent processes. As a consequence, from the results in [60], a search that arrives at a router $j \neq i$ sees the corresponding RC counter in equilibrium. Then we can immediately write,

$$\tilde{R}(t|j) \;=\; (1-\pi)^{j+1} \tag{5.9}$$

It is interesting to observe that equations (5.8) and (5.9) are identical, although reached from different assumptions. It remains to uncondition on J, the number of steps taken by the searcher by t.

We first assume, as in model 1, that the search takes an exponentially distributed random delay at each hop, independently on the system state. Second, we assume that the number of cache-routers that can be visited by the random walker after a cache miss occurs is large compared to the expected number of cache-routers that are checked by t.

Proposition 5.3 (Statefull search)**.** *The probability $\tilde{R}(t)$ that the content is not found by a request for a tagged content by time t if we use statefull search is given by:*

$$\tilde{R}(t) \;=\; (1-\pi)e^{-\gamma\pi t} \tag{5.10}$$

Proof: The proof is similar to that of Proposition 5.1. From equations (5.8) and (5.9) and since each search from cache to cache takes on the average $1/\gamma$ (exponentially distributed intervals), the number of visits by time t is Poisson distributed. Recall we do not revisit a cache, and let N be the maximum number of caches that can be visited.

$$\begin{aligned}
\tilde{R}(t) &= (1-\pi)\sum_{n=0}^{N}\frac{(\gamma t)^n}{n!}e^{-\gamma t}(1-\pi)^n + \sum_{n=N+1}^{\infty}\frac{(\gamma t)^n}{n!}e^{-\gamma t} \\
&= (1-\pi)\sum_{n=0}^{\infty}[(1-\pi)\gamma t]^n\frac{e^{-\gamma(1-\pi)t}}{e^{\gamma\pi t}} + \epsilon(N) \\
&= (1-\pi)e^{-\gamma\pi t} + \epsilon(N)
\end{aligned} \tag{5.11}$$

where $\epsilon(N) = \sum_{n=N+1}^{\infty}\frac{(\gamma t)^n}{n!}e^{-\gamma t}[1-(1-\pi)^n]$. If the tail of the Poisson distribution is negligible, or if π is very small, $\epsilon(N) \approx 0$. □

The validity of the large N assumption used in Proposition 5.3 can be checked by using the Normal distribution approximation for the Poisson distribution. For instance, the Poisson tail is very good for values of $N > \gamma\pi t + 4\sqrt{\lambda t}$.

According to (5.10), $\tilde{R}(\infty) = 0$. As the random walk time increases, the probability that the walker does not find the searched content approaches zero when contents are dynamically inserted and evicted from the caches while the walker traverses the network.

One advantage of the statefull models is that the expression obtained simplifies the solution of the optimization problem we formulate in Chapter 6 and, as such, facilitates the study of the existing performance tradeoffs.

5.3 Networks with Multiple Tiers

In previous sections we consider single tier networks. In what follows we extend these results to the multiple tier case. Then, in Section 5.4, we discuss the performance tradeoffs between these two cases.

Refer to Figure 3.1 and let M be the number of tiers. Let $\hat{\Lambda}_c$ be the publisher load accounting for the requests filtered at the M tiers. Let $R_{c,i}(T_{c,i})$ be the probability that a search that reaches domain i fails to find content c at that domain. The load for content c that arrives at the publishing area is given by:

$$\hat{\Lambda}_c = \Lambda_c \prod_{i=1}^{M} R_{c,i}(T_{c,i}) \tag{5.12}$$

where $\prod_{i=1}^{M} R_{c,i}(T_{c,i})$ is the probability that a request arrives at the publishing area and Λ_c is the load generated by the users for content c which are all placed at tier M. Note that replacing $R_{c,i}(T_{c,i})$ by $\tilde{R}_{c,i}(T_{c,i})$ corresponds to using model 2 instead of model 1 from previous section.

5.3.1 Average Delay

Let $D_{c,i}$ be a random variable that characterizes the delay experienced by requests for content c at domain i. Recall that $T_{c,i}$ is the maximum time a walker spends for content c in domain i. In what follows, we make the dependence of $D_{c,i}$ on $T_{c,i}$ explicit.

Recall that $R_{c,i}(t)$ depends on t, since the search for content c in domain i is limited to t time units. Then, we have (see, for instance, [61]):

$$E[D_{c,i}(T_{c,i})] = \int_0^{T_{c,i}} R_{c,i}(t)dt \tag{5.13}$$

When model 1 is used, $E[D_{c,i}(T_{c,i})]$ does not admit a simple closed form solution and must be obtained through numerical integration of (5.5). Instead, when model 2 is employed, we obtain

$$E[D_{c,i}(T_{c,i})] = (1 - \pi_{c,i}) \frac{1 - e^{-\pi_{c,i} T_{c,i}}}{\pi_{c,i} \gamma} \tag{5.14}$$

Considering M tiers, let D_c be the delay to find content c, including the time required for the publishing area to serve the request if needed. Then, $E[D_c]$ is given by:

$$E[D_c] = \left(\sum_{i=1}^{M} E[D_{c,i}(T_{c,i})] \prod_{j=i+1}^{M} R_{c,j}(T_{c,j}) \right) + \mathcal{C}(\hat{\Lambda}_c) \prod_{j=1}^{M} R_{c,j}(T_{c,j}), \tag{5.15}$$

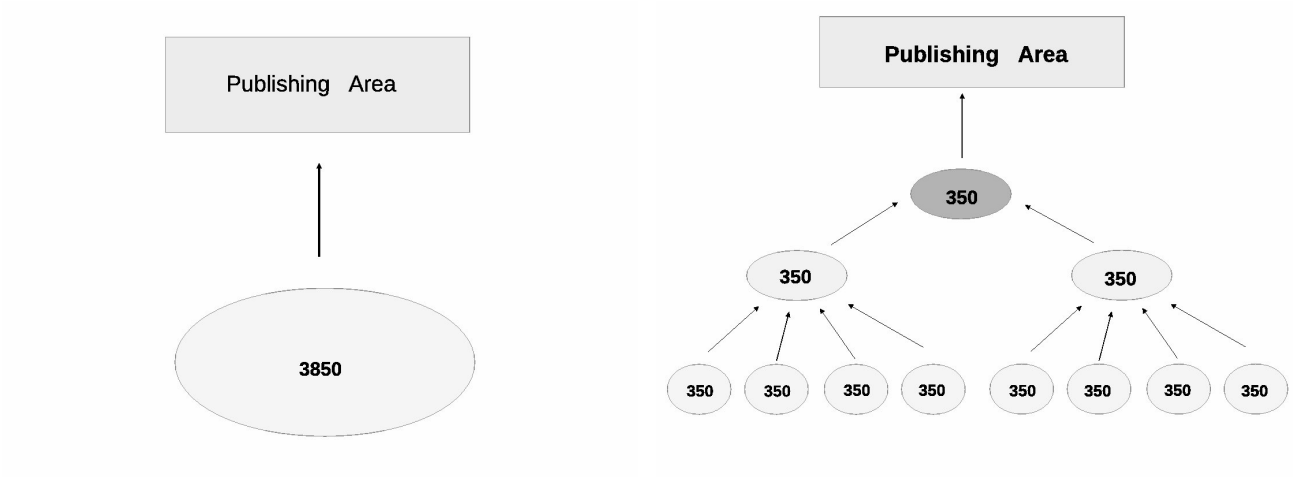
where $\mathcal{C}(\hat{\Lambda}_c)$ is the mean cost (measured in time units) to retrieve a content at the publishing area as a function of the load $\hat{\Lambda}_c$. Recall that tier 1 (resp., tier M) is the closest to the custodians (resp., users). Therefore, $\prod_{j=i+1}^{M} R_{c,j}(T_{c,j})$ corresponds to the fraction of requests to content c that reach tier i.

5.4 Results

Next, we show some results from model 1. Results from model 2 are presented in the next chapter.

We consider a single tier network with 3850 cache routers and a three tiered architecture with the same number of cache routers distributed across the tiers: 2800 in tier three, 700 in the second tier and 350 in the first tier. The tiers are divided into domains, so that the load aggregation effects can be evaluated. Both architectures are illustrated in Figure 5.1. All routers within the same tier are logically fully connected among themselves. The input load from users is modeled as a Poisson process. Random walks spend a maximum time T in each tier. The mean time to access the custodians from the publishing area, whenever a request does not find the searched content in the cache-routers, is assumed to be an exponential increasing function of the amount of requests hitting the publishing area.

We first evaluate the load that arrives at the publishing area as a function of N and T for the single-tiered scenario conditioned that the random walk is started. Figure 5.2 shows the probability that a request reaches the publishing area as a



(a) (b)

Figura 5.1: (a) 1 Tier Architecture, and (b) 3 Tiers Architecture

function of the number of routers in the domain. We consider a large value of T and we vary the number of cache-routers and ρ. The picture illustrates the following behavior: for small values of ρ it is necessary a large number of routers to obtain a small probability that a request arrives at the publishing area. This occurs because this probability can be approximated by $(1-\rho^k)^N$ as $t \to \infty$. Figure 5.3 shows the probability that a request reaches the publishing area as a function of T, the time a request spends in the tier, for $N = 3850$ (topology of Figure 5.1(a)). We note that for small values of ρ, this probability is very high and decreases very slowly with T. On the other hand, for medium and high values of ρ, this probability reaches a very low value for a small value of T.

In Figures 5.4 and 5.5 we plot the mean time to find the content for the one tiered architecture (Figure 5.1(a)) and the three tiered architecture (Figure 5.1(b)) considering four types of content popularity: very low, low, medium and high. The value of the request arrival rate for each type of content was obtained from real data collected from a major Brazilian broadband service provider.

From these plots we can observe the benefits of the load aggregation that occurs in the three tiered architecture: the requests that are not satisfied in tier three because the probability to store the content is low are aggregated in the second and third tiers and then this is increases the probability to find the content in these tiers. Recall that the mean time to retrieve the content from the publishing area is assumed to be an exponential increasing function of the amount of requests hitting

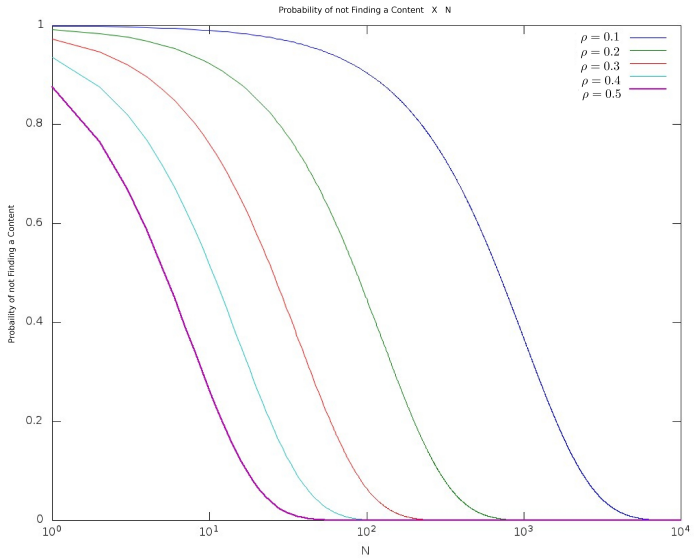


Figura 5.2: Probability that a request reaches the publishing area x N

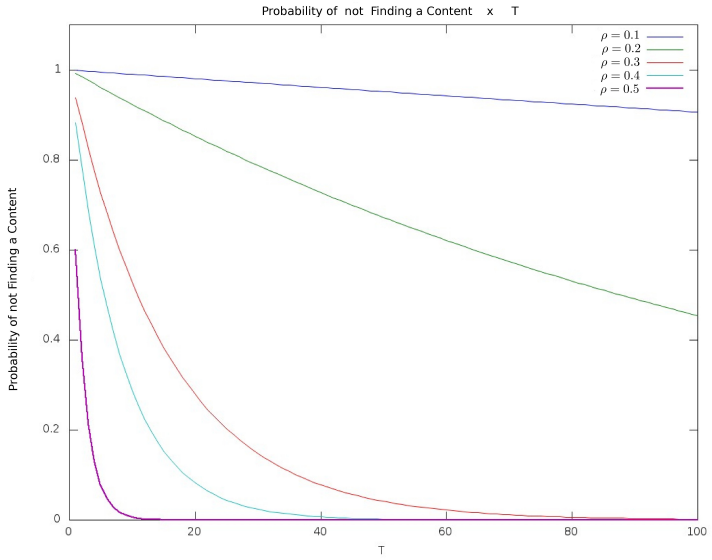


Figura 5.3: Probability that a request reaches the publishing area x T

the publishing area, then the time to retrieve the content from this area is much higher than the time to retrieve the content from one of the tiers.

We observe that the best results are achieved for low and medium popularity contents. Note that for low popularity contents the time to find content decreases by several orders of magnitude when we consider a three tiered architecture as a result of load aggregation. For very low and high popularity contents, a significant reduction is not observed. For high popularity contents, the probability to store

the content in a tier is high (for the one tiered and three tiered architectures), then only a small part of the requests are served by the publishing area. For very low popularity contents, the opposite occurs: the majority of requests are served by the publishing area because the request rate is very low and then the probability to find the content in a tier is very low (for both architectures).

In general, we can say that the three tiered architecture provides a shorter time to find the content than the one tiered architecture for all content polarities. For low and medium popularity contents, the probability to find the content in one of the tiers is high due to the load aggregation effects. For very low popularity contents, the best choice is to select $T = 0$ because the majority of requests are served by the publishing area. On the other hand, for low popularity contents, the results show that the smallest value for the mean time to find the content is achieved for $T > 0$.

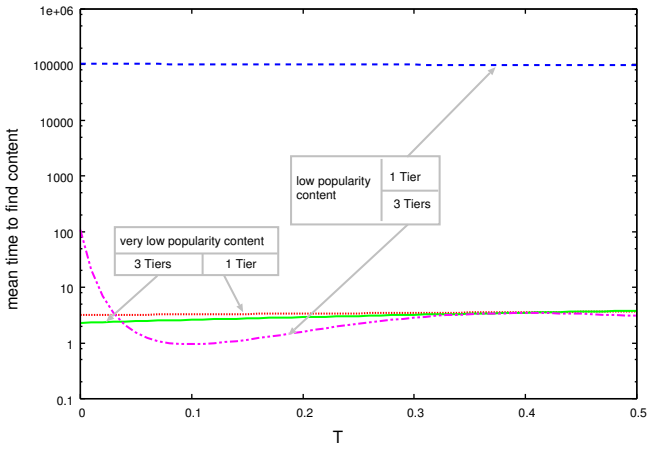


Figura 5.4: Mean time to find a content: very low and low popularity contents

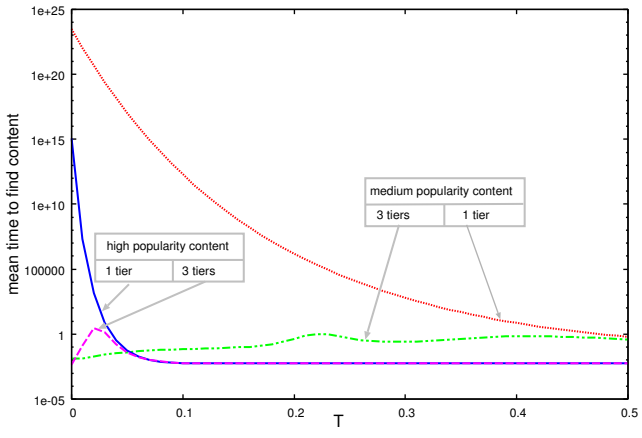


Figura 5.5: Mean time to find a content: medium and high popularity contents

We consider the one tiered topology of Figure 5.1(a). Figure 5.6 shows the probability of not finding a content in a domain for $K = 1$ and Figure 5.7 for $K = 6$. We note that the probability of not finding a content in a domain for $K = 1$ is very high for small values of T and $\rho < 0.3$, but as the value of T increases, this probability is close to zero for all values of ρ. On the other hand, for $K = 6$, even for a large value o T, the probability of not finding a content is close to one for $\rho < 0.4$. This means that if we set $K = 6$, the time to find the content can be large for low popularity contents as shown in Figure 5.9. Figure 5.8 presents the mean time of a random walk in a domain for $K = 1$ and Figure 5.9 for $K = 6$. From the figures we observe that for $K = 6$, the mean random walk time increases linearly with T for $\rho < 0.4$.

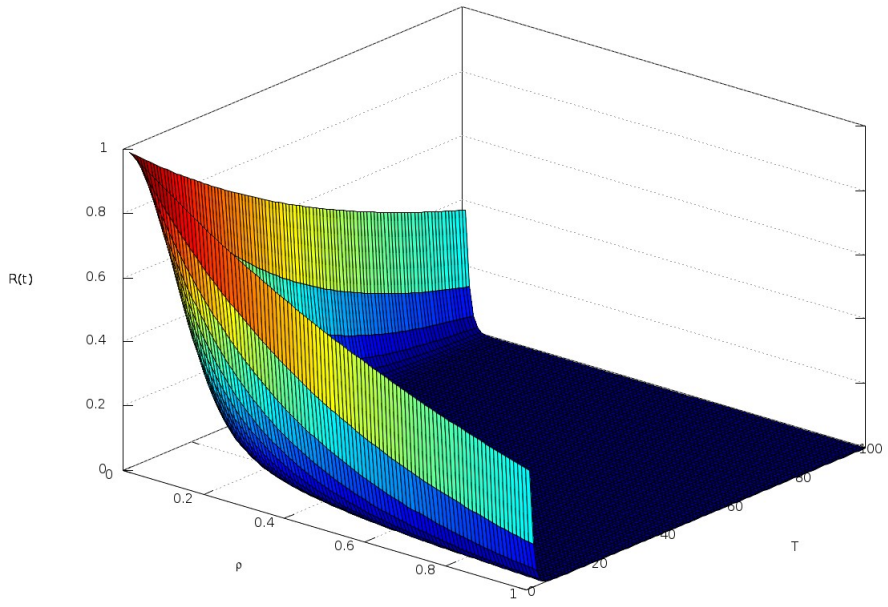


Figura 5.6: Probability of not finding a content in a domain: K=1

The analysis above shows that a small value of K provides a better performance with respect to the probability and time to find a content, which in turn will give a better performance to the user. Another important performance measure of the system are the storage requirements. The storage requirements are proportional to the fraction of time the content is in the cache. Equation 4.1 shows that the fraction of time the content is in the cache is equal to $\pi_{up}(K) = \rho^{K+1}$, then as the value of

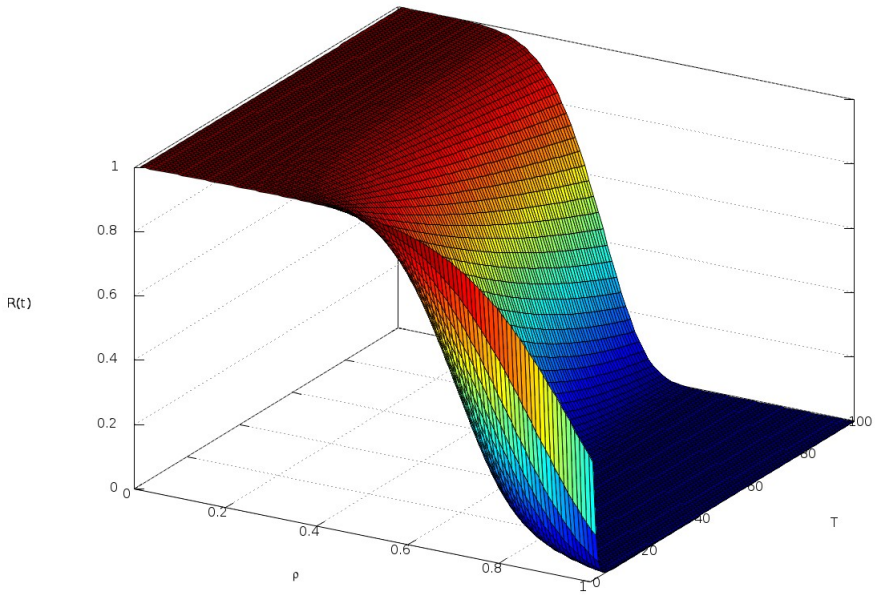


Figura 5.7: Probability of not finding a content in a domain: K=6

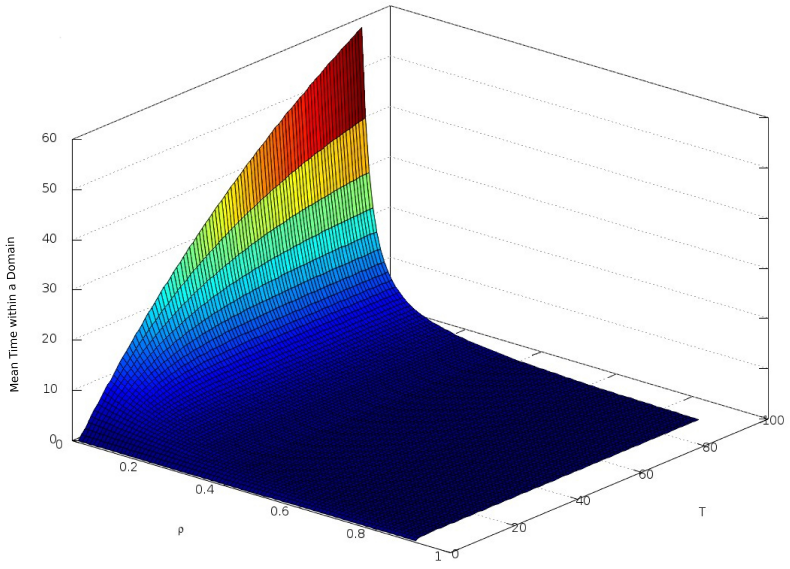


Figura 5.8: Mean time of a random walk in a domain: K=1

K increases the storage requirements decreases. So there is a tradeoff in the choice of K, we should set a small value of K to improve user performance but this will increase the storage requirements. Figure 5.10 shows the value of $\pi_{up}(K)$ for several values of ρ. We observe that for $\rho < 0.4$, the fraction of time the content is in the

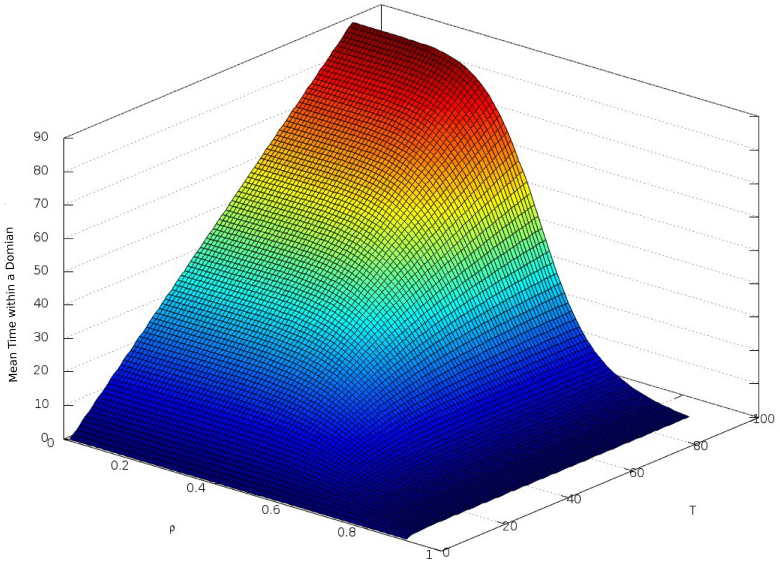


Figura 5.9: Mean time of a random walk in a domain: K=6

cache is less than 10^{-2} for $K \geq 2$. One good strategy is to set a small value for K for unpopular contents to give a better performance to the users, since this will not significantly increase storage requirements. On the other hand, if we set a small value of K for popular contents, this will greatly increase storage requirements. Figures 5.10 and 5.11 can be used to set the value of K. Figure 5.11 shows the probability of not finding the content for several values of ρ and K. From this figure one can define the value of K for a given probability and a certain content popularity. Then, using Figure 5.10 the storage requirements can be obtained.

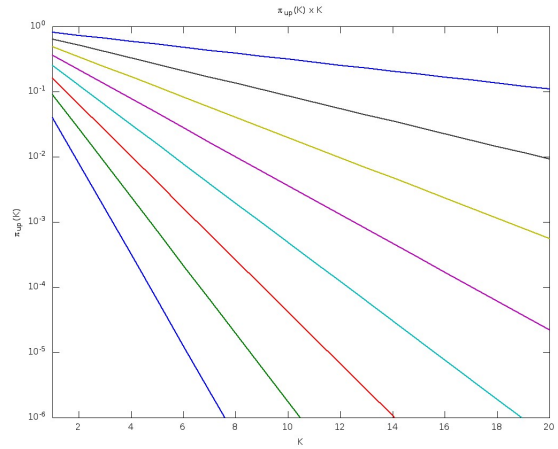


Figura 5.10: Fraction of time the content is in the cache

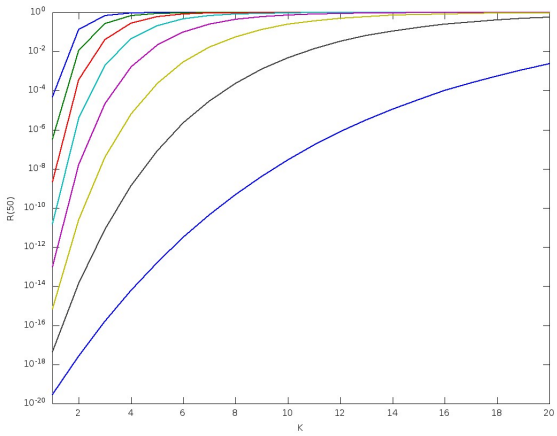


Figura 5.11: Probability of not finding a content in a domain

Capítulo 6

Statefull Model Optimization

In this chapter we consider the problem of minimizing average delay under average storage constraints. To this aim, we use model 2 that was introduced in the previous chapter. While in previous chapter the analysis targeted a single tagged content, in this chapter we account for the limiting space available in the caches, that is, when contents compete for memory space.

To simplify presentation, we consider a single tier ($M = 1$). We assume that the delay experienced by a request towards the custodian is given and fixed, equal to $\mathcal{C}$. (Additional cost functions may be used.)

Let D_c be the delay experienced by requesters of content c. $E[D_c]$ is given by (5.15) and (5.10).

$$E[D_c] = (1 - \pi_c) \left(\frac{1 - e^{-\gamma \pi_c T_c}}{\pi_c \gamma} + \mathcal{C} e^{-\gamma \pi_c T_c} \right) \tag{6.1}$$

and

$$E[D] = \sum_{c=1}^{C} \frac{\lambda_c}{\lambda} E[D_c] \tag{6.2}$$

Let $\alpha_c = 1/\mu_c$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_C)$ and $\boldsymbol{T} = (T_1, T_2, \ldots, T_C)$. In the remainder of this section, we further assume $K = 0$. In light of (4.1) and (6.1)-(6.2), we pose the following joint placement and routing optimization problem:

$$\min_{(\boldsymbol{\alpha}, \boldsymbol{T})} \quad E[D] = \sum_{c=1}^{C} \frac{\lambda_c}{\lambda} (1 - \lambda_c \alpha_c) \left(\frac{1 - e^{-\gamma \lambda_c \alpha_c T_c}}{\lambda_c \alpha_c \gamma} + \mathcal{C} e^{-\gamma \lambda_c \alpha_c T_c} \right) \tag{6.3}$$

$$s.t. \quad \sum_{c=1}^{C} \lambda_c \alpha_c = B \tag{6.4}$$

The reinforced counter vector of control variables $\boldsymbol{\alpha}$ impacts content placement, since we assume that the limited buffer size B in cache is modeled by considering

the expected number of contents in the cache ($\sum_{c=1}^{C} \lambda_c \alpha_c$), while the random walk vector $\boldsymbol{T}$ impacts content search. By jointly optimizing for placement and search parameters, under storage constraints, we obtain insights about the interplay between these two fundamental mechanisms.

In what follows, we do not solve this joint optimization problem. Instead, to simplify the solution, we solve for two independent problems: the optimal routing and then the optimal placement.

6.1 Optimal Routing Given Placement

To illustrate the insights that can be obtained with the formulated optimization problem, we consider the special case where $\alpha_c = \alpha$, for all $c = 1, \ldots, C$.

$$\alpha = B / \sum_{c=1}^{C} \lambda_c \tag{6.5}$$

and

$$\mu = \sum_{c=1}^{C} \lambda_c / B \tag{6.6}$$

$$\pi_c = \lambda_c \alpha = \lambda_c B / \lambda \tag{6.7}$$

Then, the problem reduces to

$$\min_{\boldsymbol{T}} \quad \sum_{c=1}^{C} \frac{\lambda_c}{\lambda} (1 - \pi_c) \left(\frac{1 - e^{-\gamma \pi_c T_c}}{\gamma \pi_c} + \mathcal{C} e^{-\gamma \pi_c T_c} \right) \tag{6.8}$$

$$s.t. \quad T_c \geq 0, c = 1, \ldots, C \tag{6.9}$$

For each content c the function to be minimized is $f(T)$,

$$f(T) = \frac{1}{\gamma \pi_c} \left(1 - e^{-\gamma \pi_c T} \right) + \mathcal{C} e^{-\pi_c \gamma T} \tag{6.10}$$

and

$$\frac{df(T)}{dT} = e^{-\gamma \pi_c T} - \gamma \pi_c \mathcal{C} e^{-\gamma \pi_c T} \tag{6.11}$$

Then, for a given content c, a random walk search should be issued with $T = \infty$ if $df(T)/dT < 0$, i.e., if $1 - \gamma \pi_c \mathcal{C} < 0$. Otherwise, the request for content c should be sent directly to the publishing area.

$$T_c = \begin{cases} \infty, & \pi_c > \frac{1}{\mathcal{C}\gamma} \\ 0, & \text{otherwise} \end{cases} \tag{6.12}$$

6.2 Special Case: Large γT

In the remainder of this section, we assume that γT is large. Recall that the optimization problem is given by

$$\min_{\boldsymbol{\pi}} \quad E[D] = \sum_{c=1}^{C} \frac{\lambda_c}{\lambda}(1-\pi_c)\left(\frac{1-e^{-\gamma\pi_c T}}{\pi_c\gamma} + e^{-\gamma\pi_c T}\mathcal{C}\right) \tag{6.13}$$

$$s.t. \quad \sum_{c=1}^{C} \pi_c = B \tag{6.14}$$

Let β be a Lagrange multiplier. Then, the Lagrange function is given by,

$$\mathcal{L}(\boldsymbol{\pi}, \beta) = \sum_{c=1}^{C} \frac{\lambda_c}{\lambda}\frac{(1-\pi_c)}{\pi_c\gamma} + \beta\left(\sum_{c=1}^{C}\pi_c - B\right) \tag{6.15}$$

Setting the derivative of the Lagrangian with respect to π_c equal to zero and using (6.14) we obtain,

$$\beta = \frac{\left(\sum_{c=1}^{C}\sqrt{\lambda_c}\right)^2}{\gamma\lambda B^2} \tag{6.16}$$

Therefore,

$$\pi_c = B\frac{\sqrt{\lambda_c}}{\left(\sum_{c=1}^{C}\sqrt{\lambda_c}\right)}, c = 1, \ldots, C \tag{6.17}$$

When $B = 1$, the optimal policy (6.17) is the square-root allocation proposed by Cohen and Shenker [62].

6.3 Special Case: $T = 0$

For $T = 0$, the optimization problem reduces to

$$\min_{\boldsymbol{\pi}} \quad E[D] = \sum_{c=1}^{C} \frac{\lambda_c}{\lambda}(1-\pi_c)\mathcal{C} \tag{6.18}$$

$$s.t. \quad \sum_{c=1}^{C} \pi_c = B \tag{6.19}$$

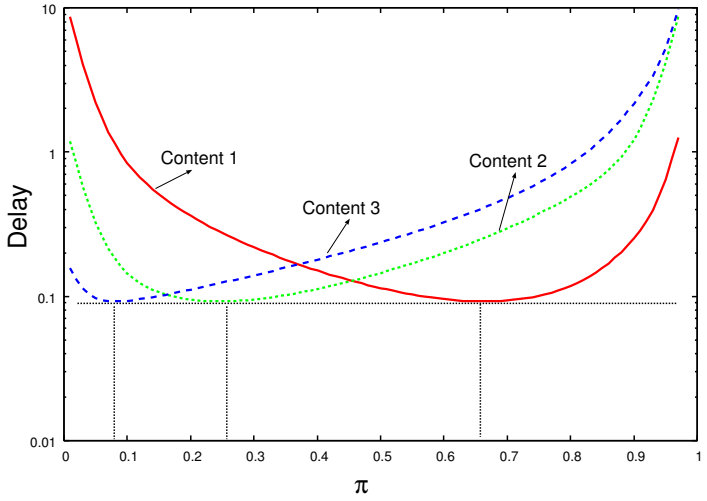
In this case, the optimal solution consists of ordering contents based on λ_c and storing the B most popular in the cache, i.e., $\pi_c = 1$ for $c = 1, \ldots, B$ and $\pi_c = 0$ otherwise. Note that this intuitive rule was shown to be optimal by Liu, Nain, Niclausse and Towsley [63].

6.4 Results

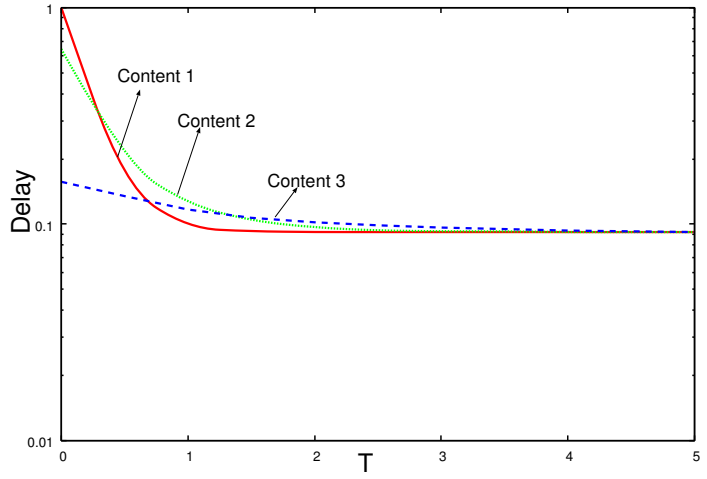
In this section we report numerical results obtained using model 2. Our goal is to obtain the values of π_c and T_c, $c = 1, 2, 3$, for which the total expected delay is minimized, and compare to those obtained from the optimization solution. We consider three contents with high, medium and low popularity sharing a memory with capacity to store, on average, one replica of the content, $B = 1$. The publisher cost $\mathcal{C}$ is equal to 10s, the random search rate is $\gamma = 10$ cache-routers/s, $\lambda_1 = 11.57$, $\lambda_2 = 1.57$ and $\lambda_3 = 0.12$. From equation 6.2, we compute the total expected delay considering all possible values of π_c and T_c for π_c varying from 0.01 to 1 and T_c varying from 0 to 5, $i = 1, 2, 3$.

Figure 6.1(a) shows the minimum delay obtained for π_1, π_2 and π_3 and Figure 6.1(b) shows the minimum delay obtained for T_1, T_2 and T_3, considering all possible values of the other parameters.

In Section 6.3, we obtain equation (6.17) to compute the optimal values of π_c. If we compute the values of π_c using equation (6.17) with the parameters of this scenario, we obtain $\pi_1 = 0.68$, $\pi_2 = 0.25$ and $\pi_3 = 0.07$. We note that these values are very similar to that of Figure 6.1(a).



(a) Minimum expected delay is obtained for $\pi_1 = 0.67$, $\pi_2 = 0.25$ and $\pi_3 = 0.08$



(b) Minimum expected delay is obtained for $T_1 \geq 2.5$, $T_2 \geq 5$ and $T_3 \geq 5$

Figura 6.1: Minimum expected delay for each value of π_c and T_c.

Capítulo 7

Conclusions

Information centric networks are gaining considerable attention from researchers and practitioners due to their scalability and robustness properties. Today's focus relies on increasing demand for multimedia files. Content Distribution Networks (CDNs) and Peer to Peer Networks (P2P) were first attempts of overlay networks to cope with content centric network issues. Nonetheless, those overlay solutions still present problems and a shift in paradigm has become necessary. Therefore, Information Centric Networks have emerged as an area of research, with important challenges in design, modeling and analysis, coupled with the deployment of a universal caching strategy. From the design perspective, one of the challenges is to determine how to provide scalability and reliability. From the modeling and analysis perspective, the challenges are associated to the large number of routers envisioned in ICNs.

In this thesis, we propose a novel design for ICNs that is built on top of random walks and hierarchical tiers (domains) to cope with ICNs open issues in general. In particular, our design addresses the exploration versus exploitation tradeoff involved in content search. Our model computes metrics such as mean time to find a content and evaluate existing tradeoffs to tune the parameteres of the architecture we propose.

We present mathematical models to study the tradeoffs between the delay spent on searching for content and the total load reaching the publishing area. We propose a novel mechanism for content placement and evaluate its performance. Our analytical model yields closed form expressions for the entire network.

Our proposal comprises a coupled policy between search and placement and we

bring light for optimization aspects considering parameters tunning. This work also opens additional avenues for future research. One such problem is to determine the impact of parallel random walks across tiers at the same level of the hierarchy as well as across different levels.

Referências Bibliográficas

[1] KUROSE, J., "Information-centric networking: The evolution from circuits to packets to content", *Computer Networks*, v. 66, pp. 112–120, 2014.

[2] "Youtube", 2015, `http://www.youtube.com`.

[3] "Wikipedia", 2015, `http://www.wikipedia.org`.

[4] "Netflix", 2014, `http://www.netflix.com`.

[5] DE SOUZA E SILVA, E., LEÃO, R. M. M., SANTOS, A. D., et al., "Multimedia Supporting Tools for the CEDERJ Distance Learning Initiative applied to the Computer Systems Course", , pp. 1–112006.

[6] "Akamai", 2015, `http://www.akamai.com`.

[7] "Limelight", 2015, `http://www.limelightnetworks.com`.

[8] "Bittorrent Project", 2015, `http://www.bittorrent.com`.

[9] "Emule Project", 2015, `http://www.emule-project.net/`.

[10] DE SOUZA E SILVA, E., LEÃO, R. M. M., MENASCHÉ, D. S., et al., "On the Interplay between Content Popularity and Performance in P2P Systems", *QEST*, pp. 27–30, 2013.

[11] CISCO, "Cisco Visual Networking Index, Forecast and Methodology, 2012 - 2017", *White Paper*, 2013.

[12] CAROFIGLIO, G., MORABITO, G., MUSCARIELLO, L., et al., "From content delivery today to information centric networking", *Computer Networks*, 2013.

[13] DE BRITO, G. M., VELLOSO, P. B., MORAES, I. M., *Information Centric Networks: A New Paradigm for the Internet*. 1st ed. Wiley, 2013.

[14] AHLGREN, B., DANNEWITZ, C., IMBRENDA, C., et al., "A survey of information-centric networking", *Communications Magazine, IEEE*, v. 50, n. 7, pp. 26–36, July 2012.

[15] XYLOMENOS, G., VERVERIDIS, C., SIRIS, V., et al., "A Survey of Information-Centric Networking Research", *Communications Surveys Tutorials, IEEE*, v. 16, n. 2, pp. 1024–1049, Second 2014.

[16] "ICN Research Challenges - ICNRG/IRTF", 2015, `https://tools.ietf.org/pdf/draft-irtf-icnrg-challenges-02.pdf`.

[17] CHOI, J., HAN, J., CHO, E., et al., "A Survey on content-oriented networking for efficient content delivery", *Communications Magazine, IEEE*, v. 49, n. 3, pp. 121–127, March 2011.

[18] PASSARELLA, A., "Review: A Survey on Content-centric Technologies for the Current Internet: CDN and P2P Solutions", *Comput. Commun.*, v. 35, n. 1, pp. 1–32, Jan. 2012.

[19] ARIANFAR, S., NIKANDER, P., OTT, J., "On Content-centric Router Design and Implications". In: *Proceedings of the Re-Architecting the Internet Workshop*, *ReARCH '10*, pp. 5:1–5:6, ACM: New York, NY, USA, 2010.

[20] DIALLO, M., FDIDA, S., SOURLAS, V., et al., "Leveraging Caching for Internet-Scale Content-Based Publish/Subscribe Networks". In: *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–5, June 2011.

[21] ANAND, A., GUPTA, A., AKELLA, A., et al., "Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination", *SIGCOMM Comput. Commun. Rev.*, v. 38, n. 4, pp. 219–230, Aug. 2008.

[22] ANAND, A., SEKAR, V., AKELLA, A., "SmartRE: An Architecture for Coordinated Network-wide Redundancy Elimination", *SIGCOMM Comput. Commun. Rev.*, v. 39, n. 4, pp. 87–98, Aug. 2009.

[23] KOPONEN, T., CHAWLA, M., CHUN, B.-G., et al., "A Data-oriented (and Beyond) Network Architecture", *SIGCOMM Comput. Commun. Rev.*, v. 37, n. 4, pp. 181–192, Aug. 2007.

[24] LAGUTIN, D., VISALA, K., TARKOMA, S., "Publish/Subscribe for Internet: PSIRP Perspective". In: *Emerging Trends from European Research, (Valencia FIA book 2010)*, 2010.

[25] "FP7 PURSUIT project", http://www.fp7-pursuit.eu/PursuitWeb/.

[26] "NETINF", 2010, `http://www.4ward-project.eu/`.

[27] "FP7 SAIL project", http://www.sail-project.eu/.

[28] GANESAN, P., GUMMADI, K., GARCIA-MOLINA, H., "Canon in G major: designing DHTs with hierarchical structure". In: *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pp. 263–272, 2004.

[29] D'AMBROSIO, M., DANNEWITZ, C., KARL, H., et al., "MDHT: A Hierarchical Name Resolution Service for Information-centric Networks". In: *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, *ICN '11*, pp. 7–12, ACM: New York, NY, USA, 2011.

[30] DANNEWITZ, C., D'AMBROSIO, M., VERCELLONE, V., "Hierarchical DHT-based Name Resolution for Information-centric Networks", *Comput. Commun.*, v. 36, n. 7, pp. 736–749, April 2013.

[31] KATSAROS, K. V., FOTIOU, N., VASILAKOS, X., et al., "On Inter-domain Name Resolution for Information-centric Networks". In: *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*, *IFIP'12*, pp. 13–26, Springer-Verlag: Berlin, Heidelberg, 2012.

[32] JOKELA, P., ZAHEMSZKY, A., ESTEVE ROTHENBERG, C., et al., "LIPSIN: Line Speed Publish/Subscribe Inter-networking", *SIGCOMM Comput. Commun. Rev.*, v. 39, n. 4, pp. 195–206, Aug. 2009.

[33] JACOBSON, V., SMETTERS, D. K., THORNTON, J. D., et al., "Networking Named Content". In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, *CoNEXT '09*, pp. 1–12, ACM: New York, NY, USA, 2009.

[34] ROSENSWEIG, E., KUROSE, J., TOWSLEY, D., "Approximate Models for General Cache Networks". In: *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, March 2010.

[35] CHIOCCHETTI, R., ROSSI, D., ROSSINI, G., et al., "Exploit the Known or Explore the Unknown?: Hamlet-like Doubts in ICN". In: *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, *ICN '12*, pp. 7–12, ACM: New York, NY, USA, 2012.

[36] WANG, L., BAYHAN, S., OTT, J., et al., "Pro-Diluvian: Understanding Scoped-Flooding for Content Discovery in Information-Centric Networking". In: *Proceedings of the 2Nd International Conference on Information-Centric Networking*, *ICN '15*, pp. 9–18, ACM, 2015.

[37] CHAI, W. K., HE, D., PSARAS, I., et al., "Cache "less for more" in information-centric networks (extended version)", *Comput. Commun.*, v. 36, n. 7, pp. 758 – 770, April 2013.

[38] FAYAZBAKHSH, S. K., LIN, Y., TOOTOONCHIAN, A., et al., "Less Pain, Most of the Gain: Incrementally Deployable ICN", *SIGCOMM Comput. Commun. Rev.*, v. 43, n. 4, pp. 147–158, Aug. 2013.

[39] WANG, Y., LI, Z., TYSON, G., et al., "Optimal cache allocation for Content-Centric Networking". In: *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pp. 1–10, Oct 2013.

[40] PSARAS, I., CHAI, W. K., PAVLOU, G., "Probabilistic In-network Caching for Information-centric Networks". In: *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, *ICN '12*, pp. 55–60, ACM: New York, NY, USA, 2012.

[41] DRESSLER, F., AKAN, O. B., "A Survey on Bio-inspired Networking", *Comput. Netw.*, v. 54, n. 6, pp. 881–900, April 2010.

[42] DRESSLER, F., *Self Organizatio in Sensor and Actor Networks*. Wiley, 2007.

[43] MITZENMACHER, M., UPFAL, E., *Probability and Computing*. 3rd ed. Cambridge Press, 2007.

[44] MOTWANI, R., RAGHAVAN, P., *Randomized Algorithms*. 1st ed. Cambridge Press, 1995.

[45] CODLING, E. A., PLANK, M. J., BENHAMOU, S., "Random walk models in biology". In: *Journal of Royal Society*, 2008.

[46] MORTERS, P., PERES, Y., *Brownian Motion*. 1st ed. Cambridge Press, 2010.

[47] RUDNICK, J., GASPARE, G., *Elements of the Random Walk: An introduction for Advanced Students and Researchers*. 1st ed. Cambridge Press, 2004.

[48] LOVÁSZ, L., "Random Walks on Graphs: A Survey", 1993.

[49] BERENBRINK, P., COOPER, C., ELSÄSSER, R., et al., "Speeding Up Random Walks with Neighborhood Exploration". In: *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, *SODA '10*, pp. 1422–1435, Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2010.

[50] RANDLES, M., ABU-RAHMEH, O., JOHNSON, P., et al., "Biased random walks on resource network graphs for load balancing". In: *Journal of Supercomputing*, 2010.

[51] DORIGO, M., STUTZLE, T., *Ant Colony Optimization*. MIT Press, 2004.

[52] PICARD, D., REVEL, A., CORD, M., "An Application of Swarm Intelligence to Distributed Image Retrieval", *Inf. Sci.*, v. 192, pp. 71–81, June 2012.

[53] CHEN, W.-N., ZHANG, J., "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements", *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, v. 39, n. 1, pp. 29–43, Jan 2009.

[54] ZHANG, J., CHUNG, H.-H., LO, A.-L., et al., "Extended Ant Colony Optimization Algorithm for Power Electronic Circuit Design", *Power Electronics, IEEE Transactions on*, v. 24, n. 1, pp. 147–162, Jan 2009.

[55] FOFACK, N. C., NAIN, P., NEGLIA, G., et al., "Performance evaluation of hierarchical TTL-based cache networks", *Computer Networks*, v. 65, pp. 212 – 231, 2014.

[56] BADAM, A., PARK, K., PAI, V. S., et al., "HashCache: Cache Storage for the Next Billion." In: *NSDI*, v. 9, pp. 123–136, 2009.

[57] PERINO, D., VARVELLO, M., "A reality check for content centric networking". In: *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pp. 44–49, 2011.

[58] DE SOUZA E SILVA, E., MUNTZ, R. R., *Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação*. SBC - Escola de Computação, 1992.

[59] MENDONÇA, G., *Residential nano cashe systems for video distribution (in Portuguese)*, Master's Thesis, COPPE/UFRJ, 2015.

[60] ROSENKRANTZ, W., SIMBA, R., "Some theorems on conditional PASTA: a stochastic integral approach", *Operations Research Letter*, v. 11, pp. 173–177, 1992.

[61] DE SOUZA E SILVA, E., GAIL, H. R., "Transient Solutions for Markov Chains", In: *Computational Probability*, pp. 44–79, Kluwer, 2000.

[62] COHEN, E., SHENKER, S., "Replication strategies in unstructured peer-to-peer networks". In: *ACM SIGCOMM Computer Communication Review*, v. 32, n. 4, pp. 177–190, 2002.

[63] LIU, Z., NAIN, P., NICLAUSSE, N., et al., "Static caching of Web servers". In: *Photonics West'98 Electronic Imaging*, pp. 179–190, 1997.

[64] REXFORD, J., DOVROLIS, C., "Future Internet Architecture: Clean-slate Versus Evolutionary Research", *Commun. ACM*, v. 53, n. 9, pp. 36–40, Sept. 2010.

[65] PAN, J., PAUL, S., JAIN, R., "A survey of the research on future internet architectures", *Communications Magazine, IEEE*, v. 49, n. 7, pp. 26–36, July 2011.

[66] WENDELL, P., FREEDMAN, M. J., "Going Viral: Flash Crowds in an Open CDN". In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, *IMC '11*, pp. 549–558, ACM: New York, NY, USA, 2011.

[67] DE SOUZA E SILVA, E., GAIL, H. R., "The Uniformization Method in Performability Analysis", In: *Performability Modelling: Techniques and Tools*, chap. 3, pp. 31–58, Wiley, 2001.

[68] DABIRMOGHADDAM, A., BARIJOUGH, M. M., GARCIA-LUNA-ACEVES, J., "Understanding Optimal Caching and Opportunistic Caching at "the Edge"of Information-centric Networks". In: *Proceedings of the 1st International Conference on Information-centric Networking*, *ICN '14*, pp. 47–56, ACM: New York, NY, USA, 2014.

[69] DOS SANTOS MENDONÇA, G. G. B., "Sistema de Nano Caches Residenciais para Distribuição de Vídeo", 2015.