Content Management in Mobile Wireless Networks

Francisco de Meneses Neves Ramos dos Santos, Chadi Barakat, Thrasyvoulos Spyropoulos, Thierry Turletti

Content Management in Mobile Wireless Networks

August 17, 2012

Supervisors: Dr. Chadi Barakat, INRIA
Prof. Matthias Grossglauser, EPFL
Dr. Thrasyvoulos Spyropoulos, EURECOM
Dr. Thierry Turletti, INRIA

Student: Francisco Neves dos Santos

École Polytechnique Fédérale de Lausanne

In memory of my Grandfather,
Eng. Manuel João das Neves.

Contents

Acronyms

List of abbreviations.

ACK	Acknowledgement
ASF	Atom syndication format
CCA	Clear-channel assessment
CCN	Content-centric network
CO	Content object
CODA	Content optimal-delivery algorithm
CTS	Clear to send
CS	Content store
CW	Contention window
DCF	Distributed coordination-function
DIFS	Distributed inter-frame spacing
DR	Delivery rate
DTN	Delay-tolerant network
EIFS	Extended inter-frame spacing
FIB	Forwarding-information base
GBSD	Global knowledge-based scheduling and drop
HBSD	History-based scheduling and drop
HM	Hello message
IM	Interest message
MAC	Medium-access control
NAV	Network allocation-vector
NTP	Normalized throughput
PCF	Point coordination-function
PIM	Pending-interests message
PIT	Pending-interests table
PHY	Physical (layer)
RTS	Request to send
RV	Random variable
SIFS	Short inter-frame spacing

TFT	Tit for tat
TTL	Time to live

Symbols

List of symbols.

$p^{(i)}$	Probability that a node receives a replica of content i
$q^{(i)}$	Request rate of content i
$\hat{q}^{(i)}$	Estimator for the request rate of content i
$n^{(i)}$	Number of replicas of content i
$\mathrm{DR}^{(i)}$	Delivery rate of content i
$\hat{\mathrm{DR}}^{(i)}$	Estimator for the delivery rate of content i
$\mathrm{MR}^{(i)}$	Miss rate of content i
$U^{(i)}(\mathrm{DR})$	Delivery-rate utility of content i
TTL	Average information-request lifetime
B	Buffer capacity per node
L	Number of network nodes
T	Total simulation time
R	Number of contents requested by a node per unit of time
b_{hello}	Backoff counter for computing the delay between hello messages
b_{PIM}	Backoff counter for computing the delay between pending-interests messages
d_{hello}	Transmission delay of the next hello message
$d_{\mathrm{hello}}^{\mathrm{min}}$	Minimum delay between hello messages
d_{PIM}	Transmission delay of the next pending-interests message
$d_{\mathrm{PIM}}^{\mathrm{min}}$	Minimum delay between pending-interests messages
α	Weight given to the most recent request-rate estimate
β	Weight given to the most recent delivery-rate estimate
γ	Exponent of the Zipf distribution of content popularity
λ	$1/\lambda$ is the average node inter-meeting time
ς	Lagrange multiplier

Chapter 1
Introduction

Abstract Approximately one billion users have access to mobile broadband, through which they intend to obtain the same data they can reach using a wired connection. Because of the cost of transmitting data over a mobile-broadband connection and given that 3G networks are quickly reaching their data-transfer capacity, some researchers envision the inter-connection of mobile devices using Wi-Fi, forming a challenged network. Such networks suffer from high latency, low data rates, and frequent disconnections; because end to end paths between pairs of nodes may not always exist, a mobile device must store content before delivering it to the intended receivers. We designed the content-optimal delivery algorithm (CODA) for distributing named data over a delay-tolerant network (DTN), which is a network of challenged networks. Current content-dissemination techniques for DTNs consist mainly of the following items: a content store, for caching and indexing retrieved content, and a query and response mechanism to search the network for matching content. Some algorithms attempt to optimize an objective function, such as the total delivery-delay. While disseminating content, CODA maximizes the network throughput by computing the utility of each item published: a device with a full buffer drops content in order of increasing utility and transmits content in order decreasing utility. We implemented CODA over the CCNx protocol, which provides the basic tools for querying, caching, and transmitting content.

1.1 Problem Statement

The Internet, drafted in the 60s, was intended as a means to share the access to mainframe computers among users, enabling them to perform complex computations remotely [11][19]; it supports the communication between two hosts, a source and a target, each referenced by a network address that is stored in the header of an IP packet [19]. The emergence of the World Wide Web and the reduction in computer prices changed the way we benefit from the Internet [9][19]. Fifty years since the advent of the Internet, the sharing of data from single hosts to multiple recipients

is the principal contributor to IP traffic growth [30]. Users want to access data, such as text, sound, or video files, without worrying about its location [19]. The difficulty in linking content to IP addresses using the existing Internet architecture lead to the emergence of the **content-centric networks** (CCN) [19]. A CCN implements an alternative communication model whereby a user requests data by name and any node with matching content can respond with the appropriate data [19][25]; CCN nodes replicate content amongst themselves in order to improve the network's ability to satisfy user requests.

An estimated 1 billion people worldwide subscribe to mobile-broadband [20][18], through which they intend to access the same data they can obtain using a wired connection. However, because of the high bandwidth required for sharing data, especially video, this option can be costly [32]. Additionally, 3G networks are quickly reaching their data-transfer capacity [10][25]. To avoid these pitfalls and because many phones feature WiFi connectivity, some researchers advocate the direct connection of mobile devices using wireless standards [25], such as the IEEE 802.11 [14], to form a **challenged network** [8]. In such networks, end-to-end paths between any two nodes may not always exist, forcing nodes to store content before delivering it to the intended receivers, and data transmissions occur when pairs of nodes are within range. Additionally, we envision inter-connecting these groups of devices through a delay-tolerant network. A DTN is a network of challenged-networks that support long delays and data loss between and within challenged networks [8]. In this work, we study the problem of disseminating content to multiple receivers over a delay-tolerant network [6] using memory- and battery-constrained devices. We propose the content optimal-delivery algorithm (CODA) that aims to optimize the **delivery rate** of all content exchanged in such a network, measured as the number of content replicas delivered to the intended receivers per unit of time. We assume that (i) a node moves randomly and independently of its peers, (ii) node inter-meeting times are exponentially distributed with a constant rate λ, (iii) a node may enter and leave the network at any time, and (iv) each node has a reduced buffer for storing messages.

A node operating in this network can have the following three roles: source, forwarder, and receiver. A source node generates data and wraps it as a **content object** (CO), which contains the following items: a unique name, the data's issue time, the validity, which indicates for how long the information is accurate, and the data itself. A forwarder node stores content it receives from peers and delivers it to other nodes it meets; this process produces new replicas of the original content object. Because a node has a reduced buffer space for storing content, the forwarder must choose wisely which replicas it caches in order to optimize the global delivery-rate. Additionally, a forwarder must decide which replicas to send first to a node it meets in order to improve the global delivery-rate; this is because the meeting time between both nodes might be insufficient for the forwarder to send all content replicas to its peer. A node acting as a receiver requests data through an **interest message** (IM), which identifies the name of the content it wants to obtain and the amount of time it is willing to wait for the requested data. To assess CODA's performance, we

simulate the effect of varying the storage capacity per node, network load, and the distribution of content popularity.

1.2 State of the Art

In this section we present existing solutions for delivering information to users over a delay tolerant network. Such a network suffers from frequent topology changes and, consequently, content publishers and subscribers are seldom connected simultaneously. To counter this problem, a node must replicate content received from a publisher so that the data remains available to potential receivers after the publisher disconnects. We adopted several strategies proposed in this section in developing our own algorithm for delivering content in DTNs.

ContentPlace [4] is a content-distribution system designed for opportunistic networks; each data source organizes content into channels to which several receivers can subscribe. The authors assume that users have friends in the network and exploit these social relationships in order to increase the fraction of satisfied data-requests (**hit rate**), an idea that was previously applied to message forwarding in [16]; each user is affiliated with a **home community** and can also be associated with other communities, dubbed **acquainted communities**. A node initiates a dialog with a single peer at a time; it calculates the utility of all locally-cached objects and those stored in its partner's buffer, picks the objects that maximize the utility of its own cache, downloads the requested objects from its peer's buffer, and, finally, discards from its own buffer data-items that are not part of the previous selection. Specifically, when two nodes meet, each node solves an optimization problem whose goal is to maximize the utility of the content it stores, subject to the space constraints of its own buffer. A node calculates the utility of an object by considering the value this data has according to each community it is affiliated with, multiplied by the strength of the user's bond to this community; hence, the utility equation consists of a weighted sum of the data-item's worth to each community the user is associated with. A device estimates the usefulness of an object to a community by multiplying the data's **access probability** with its **cost** and dividing by its size; intuitively, as an object permeates a community, it becomes easier to locate and the need for its replication subsides. The authors validate the proposed solution by measuring the hit rate in simulation scenarios with forty-five and ninety nodes, dispersed over a grid of width one kilometer, where each node seeks data at an average rate of 3 requests per 10 min and each device subscribes to just one data-channel. For the social strategy used in ContentPlace to produce a noticeable improvement on the hit-rate, the authors assume an unbounded delay for the interest messages. In our own work, we propose an algorithm that delivers content for higher request-rates.

The authors in [12] propose a middleware system that allows users to publish and subscribe content in opportunistic networks. The system extends the **atom syndication format** (ASF) [29] and organizes content into **feeds**, where each feed consists of multiple **entries**, possibly published by different users, and each feed can contain

multiple **enclosures**. Each feed and entry has a permanent globally unique identifier, generated by the publisher, a title and a timestamp, which indicates the time of the last update. An enclosure is a file that holds audio, video, or textual data; the middleware divides this file into small chunks of 16 kB before transmission to increase the chance of delivering the feed successfully. A node sends periodic **hello messages** to its link-layer neighbors through the discovery module; this message includes the id of the node performing the discovery, along with the revision number of its content store. The discovery module stores the revision counter for each peer met; an increment in the revision value means new content is available for download since the last synchronization. When a node discovers a peer, it sends a **request message** to solicit a specific feed or learn of existing content for download; its peer sends a **reply message** with the matching content (if available) or the set of feeds available in its cache, depending on the type of request. The middleware system uses a Bloom filter [5] to reduce the amount of space occupied by the content index and to shorten the time necessary for a subscriber to determine which feeds are currently buffered by the publisher. This type of filter is subject to false-positive errors, meaning that subscribers may find content id's in the filter that do not point to actual data in storage. Because false-negative errors cannot occur in Bloom filters [5], id's not found in the filter imply the content is not available in the publisher's buffer. A content publisher can serve multiple subscribers and a subscriber can download feeds from several subscribers in parallel. Because this content-dissemination system is purely interest driven, a node only stores and forwards content it is personally interested in. We use an identical mechanism for discovering peer devices as implemented in the discovery module; however, CODA accepts to store and forward unsolicited content in order to improve the delivery rate.

The authors in [15] consider the problem of disseminating information among wireless nodes in an ad hoc network; they define a **channel** as a source of content that generates new data periodically. Their goal is to find the optimal strategy for allocating forwarders to channels in order to reduce the overall content dissemination-time. To study this problem, they consider a system of N wireless nodes that must spread the data originating from J channels over an ad hoc network. Each node u subscribes to a fixed set of channels $S(u)$ and has a buffer capacity $C(u)$, measured in channel units; hence, u must dedicate $C(u) - |S(u)|$ storage units to help forward other network content. In the proposed framework, s_j denotes the proportion of nodes that subscribe to channel j, $f_j(x)$ represents the fraction of nodes that forward channel j, and each channel j has a utility function $U_j(t_j)$ that quantifies the satisfaction of a subscriber of j given the dissemination time t_j and is a non-increasing function of t_j. In the same framework, $V_j(f_i) = U_j(t_j(f_j))$ expresses the utility of channel j with respect to the fraction of forwarders f_j; as the portion of channel forwarders increases, content is delivered faster, and $V_j(f_i)$ increases.

The chosen framework [15] uses a standard definition of system welfare given by the weighted sum of the channel utilities, for positive weights $w_1, \ldots, w_J$. The authors analyze two forms of the system welfare: (i) the **channel-centric** form, where each channel has a weight equal to one, and (ii) the **user-centric** form, where the weights are equal to the share of users forwarding each channel. To find the

optimal allocation of forwarders to channels, they [15] use a centralized greedy-algorithm that, at each iteration, finds a channel for which there is a node with enough capacity to forward it and such that the increment in the system welfare is highest; if such a channel exists, the algorithm increments the number of helpers for that channel; if no such channel is found, the algorithm outputs the channel forwarding strategy.

To assign forwarders to channels using a distributed algorithm, the authors [15] suggest the following algorithm: when two nodes, u and v, meet, node u picks a channel j uniformly at random from its set of helped channels and selects another channel j' uniformly at random from its peer's set of forwarded channels. Node u computes the probability for exchanging j for j', which is proportional to the fraction of forwarders of j with respect to portion of transmitters for j', $f_j/f_{j'}$, and to the improvement in the system's welfare. Every node u maintains an estimate of f_j, $\hat{f}_j$, for every channel j in the network. The results show that the distributed method for assigning channels to forwarders approaches the optimal solution, found using the centralized algorithm. In our own work, we use an estimation method similar to the one used for finding f_j. CODA aims to optimize the delivery rate and not the dissemination time; however, by using a utility function such as the one proposed here, we could adapt our solution to optimize the time to propagate content. One fundamental difference between CODA and this framework is that we do not assume nodes meet for a sufficiently long time to exchange useful content between themselves.

Wireless ad hoc podcasting [27] is a technique for sharing content between mobile nodes in a peer to peer fashion, based on the atom syndication format [29]; content is organized into feeds and a node solicits entries for one or more feed channels. A user's mobile device stores feeds that may be of interest to other devices, in addition to the data subscribed by the user. Each device posts the set of feeds that it currently holds through the **discovery channel**; the set is implemented as a Bloom filter, just like in [12]. The device [27] stores feeds in two separate buffers: (i) the **private cache**, which contains data subscribed by the user, and (ii) the **public cache**, which holds data intended for other users. A mobile node may act as a **content provider** (server), a **content solicitor** (client), or both roles simultaneously. When two devices meet, the client seeks data of interest to the user from its peer's discovery channel and the serving node replies to the incoming requests; after obtaining all content subscribed by the user, the client device looks for feeds to fill its public cache. There are five methods to load a device's public cache: (i) most solicited, fills the public cache with the most popular content, based on previous requests, (ii) least solicited, privileges content that is seldom requested, (iii) uniform, a device loads the cache with random-content that it has seen in the past, (iv) inverse proportional, selects feeds with a probability that is inversely proportional to the content's popularity, and (v) no caching, using this strategy, the device simply eliminates its public cache. For each strategy, they measure the overall and per channel mean-**freshness** and **fairness**; freshness is the amount of time elapsed between the receipt of a feed and its publication and fairness quantifies how well each channel is served, irrespective of its popularity. All strategies score better than

"no caching"; hence, buffering unsolicited channels improves system performance. Given that users request content according to a Zipf distribution, it is surprising that the "uniform" strategy achieves the best overall performance [27].

MobiTrade [24] is a content-dissemination system for DTNs that uses a tit for tat strategy (TFT) to penalize selfish users, which attempt to share only the content they personally consume. According to this strategy, if two nodes meet, a node trades one content for each content it receives from its peer. The reward obtained by a node a following a data-exchange with a node b is equal to the amount of content a retrieves from b. TFT encourages a node to look for content of interest to its peer in order to increase the amount of data it can gather from the data-exchange. A MobiTrade user requests a set of contents through a **channel record**, comprising a set of keywords that describe the data of interest. A node predicts the channel's demand by computing the exponentially weighted moving average of the previous demand and the sum of the current demand with a speculation component; it then allocates buffer space for each channel in proportion to the reward it expects to obtain for selling this content. The authors [24] compare the performance of MobiTrade to Podcasting [27] by measuring the average delivery-rate, defined as the amount of content received divided by the total content generated, for each channel subscribed by a user; further, they [24] quantify the gain in performance of the tit for tat strategy. The results show that MobiTrade with the TFT strategy outperforms Podcasting; however, in the presence of selfish users, the performance of MobiTrade with and without TFT is identical. We follow a similar approach as in [24], but employ a different utility function; CODA aims to optimize the delivery rate of content in the network. Further, we deploy our solution over the content-dissemination protocol CCNx [31] in order to benefit from standard procedures to request, store, and forward content.

1.3 Contribution

The authors in [24] present two message scheduling and drop policies for delay tolerant networks, GBSD and HBSD, that optimize the delivery of individual messages to single targets. The first policy (GBSD) requires complete network knowledge in order to operate while the second (HBSD) estimates network-parameters using data that is locally available to each DTN node. Both policies aim at optimizing one of two possible metrics: delivery rate and delivery delay. By optimizing with respect to the delivery rate, nodes maximize the number of messages reaching the intended destination; using the delivery-delay metric, nodes minimize the average delay for a message to reach the intended receiver. We leverage on this work to implement our own solution for transmitting data over a DTN and extend it to consider multiple targets instead of a single information-receiver.

We propose the content optimal-delivery algorithm, CODA, that distributes content to multiple receivers over a DTN. CODA assigns a utility to each content item published in the network; this value gauges the contribution of a single content replica to the network's overall delivery-rate. CODA performs buffer management

by first calculating the delivery-rate utility of each cached content-replica and then discarding the least-useful item. When an application requests content, the node supporting the application will look for the content in its cache. It will immediately deliver it to the application if the content is stored in memory. In case the request cannot be satisfied immediately, the node will store the pending request in a table. When the node meets another device, it will send the list of all pending requests to its peer; the peer device will try to satisfy this list by sending the requester all the matching content stored in its own buffer. A meeting between a pair of devices might not last long enough for all requested content to be sent. We address this problem by sequencing transmissions of data in order of decreasing delivery-rate utility. A content item with few replicas in the network has a high delivery rate utility; these items must be transmitted first to avoid degrading the content delivery-rate metric. The node delivers the requested content to the application as soon as it receives it in its buffer. We implement CODA over the CCNx protocol [31], which provides the basic tools for requesting, storing, and forwarding content.

Chapter 2
Background

Abstract In this chapter we provide background knowledge on the CCNx protocol, delay-tolerant networks, and the IEEE 802.11 wireless medium-access control protocol, which is required to understand the forthcoming sections of the report. CCNx is a transport protocol for delivering content to multiple recipients; it defines two message types: an interest message, through which a user requests content, and a content object, which comprises the requested data. Each CCNx device manages the following data structures: a content store, where it buffers retrieved data objects, a pending-interests table, which holds all unsatisfied content-queries, a forwarding-information base, comprising a list of peer addresses through which it attempts to satisfy pending interests, and a face, which denotes a generic interface for communicating with other network peers. A delay-tolerant network is a network of challenged networks, whose links are characterized by high latency, low data-rates, and recurrent disconnections. A DTN gateway provides a store and forward service between heterogeneous networks and offers a message switching interface to the upper layer protocols. The IEEE 802.11 defines two medium access protocols for the MAC layer: the point coordination function (PCF) and the distributed coordination function (DCF); we focus our analysis on the DCF access method. The DCF is based on the carrier sense multiple access with collision avoidance algorithm: a source node transmits data if it senses the medium idle, otherwise it waits a random amount of time before attempting to transmit again.

2.1 The CCNx Protocol

The CCNx protocol [31][19] is a transport protocol for delivering named content to multiple targets in content-centric networks. It defines two message types: interest messages and content objects; an interest message is used to query data by name and a content object is the data itself. The name given to a content instance is independent of its physical location and it has a hierarchical structure. A CCNx name in an interest message may target a specific chunk of data or multiple chunks, in which

case the name is a prefix of all matched content names, in the same way that an IPv4 network-address is a prefix of all member hosts [31]. A CCNx node uses the following four data structures: a **content store** (CS), which is a buffer for storing received content objects, a **forwarding information base** (FIB), containing a list of destinations for forwarding interest-messages that have no matching object in the CS, a **pending interest table** (PIT), which stores interest-messages that have been forwarded using the FIB and are pending an answer, and a **face**, which is a generic interface through which messages are sent and received.

Upon receiving an interest message, a node searches for matching content objects in the CS; if one or more objects are found, the node sends the best matching object through the requester's face. When the node fails to retrieve a matching content object from the CS, it checks if the PIT does not yet contain a copy of the interest message, forwards the interest message using the FIB, and adds an new entry to the FIB containing the requester's face; if the FIB contains a copy of the interest message, the node has already forwarded the message to other nodes, in which case the requester's face is simply added to the existing FIB entry. When a node receives a content object, it searches the CS for an existing copy; if a matching object is found, the received content is a duplicate and can be discarded. The node searches for matching interest messages in the PIT and sends the object to all faces that have requested this data. If no match is found in the PIT, the content object is unsolicited and is evicted.

The CCNx protocol provides a set of tools that are convenient for the development of our own work, such as the caching of content items, delivery of data based on information requests, removal of duplicate objects and the verification of the integrity of data objects. By implementing our algorithm over the CCNx framework, we can concentrate on managing each node's buffer space and scheduling the transmissions in order to maximize the total content delivery-rate.

2.2 Delay Tolerant Networks

A delay-tolerant network [8][6] aims to inter-connect **challenged networks**, whose links are characterized by high latency, low and (possibly) asymmetric data-rates, and frequent disconnections. Node mobility and devices with short duty-cycles are the fundamental cause of disconnections in wireless networks [8]. Motion-induced disconnections can be **predictable**, e.g. satellite motion, or **opportunistic**, such as when a node comes within communication-range of a peer device by following a random walk. Disconnections due to short duty-cycles occur frequently among low power devices, such as sensor nodes, and are often predictable. The frequent disconnections lead to long queueing delays, in the order of hours or days; furthermore, the lack of transmission opportunities makes source-driven retransmissions expensive. This implies that messages must be stored within the network for long periods of time. In some challenged networks, nodes may have a lifespan that is shorter than the time to deliver a message to a receiver; in such cases, conventional end-to-end

acknowledgments are useless and reliability must be implemented on a hop-by-hop basis.

A delay tolerant network works above existing protocol-stacks, offers a store and forward service between heterogeneous networks, and provides a convenient message-switching interface to the upper layers [8]; in this context, messages are also known as **bundles** [33] and message routers are dubbed **DTN gateways** or **bundle forwarders**. Figure 2.1 illustrates the fundamental components of the DTN architecture. A DTN gateway extending across two regions consists of two units, each unit functions above the transport protocols of each region; two nodes belong to the same **region** if they communicate without requiring a DTN gateway. Whenever a source application requires reliable delivery, the gateway must store bundles in nonvolatile storage (illustrated as "Data" in Figure 2.1). The gateway uses the name tuple in the DTN message header to route the packet to a specific region; a **name tuple** consists of two variable length sections: (i) the region name and (ii) the entity name. The region name is hierarchically structured and is globally unique; the entity name is resolvable within a specific region and is not (necessarily) globally unique. A DTN routing-protocol uses the region name to forward the message through multiple regions, interconnected by gateways, until it reaches the destination region; at this point, the last gateway translates the entity name into a protocol-standard name (or address) that can be recognized within this region. The DTN architecture does not assume the existence of a path between a source and destination; rather it expects a node to forward a message to its next-hop neighbor whenever a communication-opportunity arises. The goal of routing protocols within each region is to find the path to a destination node by exploiting such transmission opportunities and, assuming that pairwise node contacts are short-lived, choose which messages to send first to the next-hop node; examples of such algorithms include HBSD [24] and RAPID [2].

An application using a DTN disposes of three classes of service to deliver messages: bulk, normal, or expedited [33]; additionally, the source application can request a delivery receipt, confirming that the target node received the bundle. The DTN architecture supports a **reliable delivery** option that requires each node to acknowledge receipt of the message to the previous-hop node, a procedure known as **custody transfer** [8]. The DTN architecture considers two types of routers: persistent (P), which have capacity to store messages, and non-persistent (NP), which might lack caching capabilities. A persistent DTN node participates in the custody-transfer protocol, unless it is unable or unwilling; it clears expired messages from its buffer and rejects oversized bundles. When a persistent router receives custody of a message, it must store it in non-volatile memory until it delegates the delivery responsibility to the next-hop P node; it sequences the next-hop transmissions based on each message's priority and lifespan.

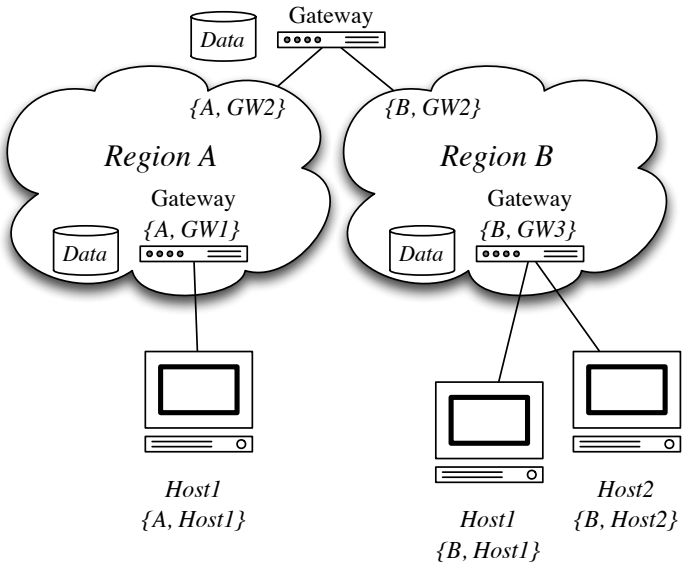


Fig. 2.1: The delay-tolerant architecture [8]. A name tuple comprises two units of variable length: {region name, entity name}; the region name is globally unique and the entity name is unique within a specific region. Host1 connects to region A through gateway $\{A, \text{GW1}\}$. The gateway with interfaces $\{A, \text{GW2}\}$ and $\{B, \text{GW2}\}$ interconnects regions A and B; it provides a store and forward service between hosts of each region. Region B contains two hosts, $\{B, \text{Host1}\}$ and $\{B, \text{Host2}\}$; note that the name "Host1" is not globally unique.

2.3 Wireless-Medium Access Control

Due to the popularity of the IEEE 802.11 wireless standard [14] among wireless devices, we analyze the performance of CODA assuming that mobile nodes communicate using the medium access control (MAC) and physical (PHY) layers defined by this standard [17][28]. A wireless network abiding to this norm has two operating modes: **infrastructure** and **ad hoc** [28]. An "infrastructure" WLAN requires an access point to act as a hub, through which mobile devices communicate; a network functioning in "ad hoc" mode does not require an access point and mobile nodes inter-connect dynamically to form an arbitrary topology. For our purposes, we assume that all mobile devices function in "ad hoc" mode.

The standard [17] defines two medium access mechanisms for the MAC layer: point coordination function (PCF) and distributed coordination function (DCF). The PCF is a centralized medium-access method whereby a base station polls each sender for data. In our network scenario, we assume that mobile devices use the DCF

method to access the wireless medium, which is based on the carrier sense multiple access with collision avoidance algorithm [28]. If a source node senses the medium busy, it chooses a random **backoff time** from the interval $[0, \mathrm{CW})$, where CW denotes the **contention window**. This backoff time is measured in slot times, where a **slot time** is sum of the receiver to transmitter turnaround time, MAC processing delay, and the clear-channel assessment (CCA) time. The source node decrements the backoff time by one for each DIFS period during which it perceives the medium to be idle; the **distributed inter-frame spacing time** (DIFS) is equal to **short inter-frame spacing time** (SIFS) plus twice the slot time: $\mathrm{DIFS} = \mathrm{SIFS} + 2(\text{slot-time})$. The wireless device freezes the timer whenever it detects activity in the medium and resumes the count down as soon as it senses the medium idle for a DIFS. The CW is doubled upon each unsuccessful transmission, up to a maximum of $\mathrm{CW}_{\max} + 1$, where $\mathrm{CW}_{\max}$ is a characteristic of the physical layer. The node transmits the data packet when the timer drops to zero and if it senses the medium idle [1]. Wireless nodes overhearing this radio communication configure their **network allocation vector** (NAV), which contains an estimate of the duration of the transmission [28]. Whenever a node recognizes an incorrect frame, it postpones its transmission for an **extended inter-frame spacing** (EIFS) period, which lasts for $\mathrm{SIFS} + \mathrm{ACK}_{\text{time}} + \mathrm{DIFS}$ seconds. Figure 2.2 describes a typical message-exchange for the basic access-method of the DCF.

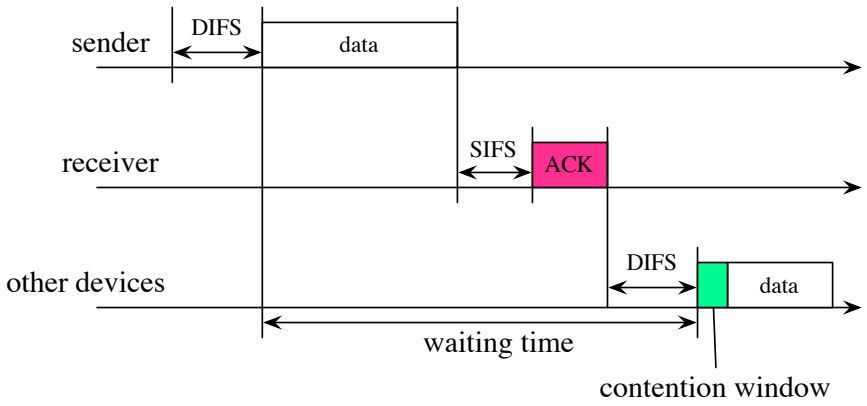


Fig. 2.2: Basic medium-access mechanism for the distributed coordination function [28]. The sender transmits a data packet after sensing the medium idle for DIFS. Upon receiving the data packet, the receiver waits for SIFS to allow the sender enough time to become a receiver; then the receiver acknowledges receipt of the data packet. Other devices wait until the medium is idle for a DIFS period and start decrementing the backoff timer before transmitting.

The optional medium-access method of the DCF requires a transmitter to reserve the wireless medium before it sends a data packet. This access mode is based on the MACAW protocol [3] and introduces two control-frames: **request to send** (RTS) and **clear to send** (CTS) [28]. A node that wants to send data, transmits a RTS frame indicating for how long it wants to reserve the channel [1]; upon receiving the RTS frame, the target device replies with a CTS, containing the channel reservation time minus the sum of the duration of the CTS message and SIFS: reservation time $-$ (CTS$_{\text{time}}$ + SIFS). A wireless terminal overhearing a RTS or CTS frame, updates its NAV to reflect the duration of the reservation and refrains from any communication during this period. Figure 2.3 illustrates a message-exchange example for the optional channel-access mechanism of the DCF. The RTS-CTS mechanism handles the **hidden terminal** problem [1]. This problem occurs when two wireless devices, *a* and *b*, cannot hear each other, but a third node, *c*, can listen to both; if terminals *a* and *b* send a packet at the same time, *c* may be unable to receive the data successfully. Using the DCF's optional mode, *a* and *b* send a RTS frame to *c* before transmitting their data packets; in turn, *c* reserves the medium for either *a* or *b*. In practice, if the RTS and CTS frames are lost, collisions can still occur at the receiver *c*. A device transmits control frames, such as RTS, CTS, and ACK frames, at the lowest possible transmission-rate to insure robustness against interference [21]; because devices transmit data at much higher data rates, the RTS and CTS frames pose a significant bandwidth-overhead [1]. For these reasons, the optional DCF mode is usually turned off.

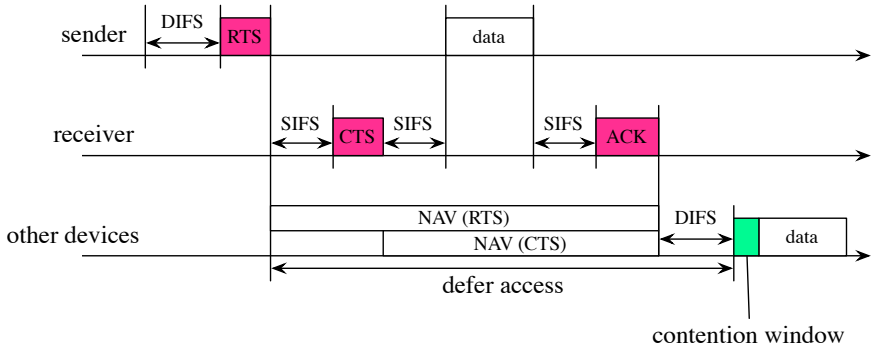


Fig. 2.3: Optional medium-access mechanism for the distributed coordination function [28]. Upon sensing the medium idle for a DIFS, the sender reserves the shared medium by sending a request to send frame. As it has received no other RTS messages, the receiver replies with a clear to send frame. Other devices overhearing the RTS and CTS messages adjust their network allocation vector and refrain from transmitting during the reserved period.

Chapter 3
Solution

Abstract The content optimal-delivery algorithm receives interest messages from the application and attempts to retrieve matching content from the device's buffer. If no corresponding content is found, CODA stores the unsatisfied request in the pending-interests table. Periodically, each device issues a hello message to signal its presence; this message prompts CODA to send a pending-interests message with the list of all unsatisfied queries. A neighboring device tries to satisfy this list by fetching content from its own cache. We introduce a mechanism that regulates the delay between hello messages to avoid congesting the network. CODA aims to maximize an objective function while disseminating content: this function measures the total delivery-rate or network throughput. For this purpose, our algorithm calculates the per-content utility, defined as the gradient of the objective function, by estimating the content's request- and delivery-rate using local network data. To optimize the network throughout, CODA endeavors to equalize the utility for all content: when a device's cache is full, CODA discards the least-useful replica first; similarly, it sequences content transmissions by decreasing utility. In order to implement CODA over CCNx, we introduce two new fields in the protocol's content object that carry the request- and delivery-rate estimates, disable the propagation of interest messages in CCNx, and introduce two control packets to implement the hello and pending-interests messages.

3.1 Designing the Solution

The content optimal-delivery algorithm distributes content to multiple users over a delay-tolerant network while maximizing the total delivery-rate (DR); the **delivery rate** is the number of satisfied content-requests per unit of time. To achieve this, we derive a utility function, $U^{(i)}(\mathrm{DR})$, that measures the marginal gain in the total DR with respect to the number of existing copies of content i in the network. A device determines the utility of a content using information that it gathers locally. We prove that if nodes use this utility function to eject less-useful content from a full buffer

and to prioritize data-transmissions, the delivery-rate utility of all content will, ultimately, be equal: $\forall_{j=1}^{k} U^{(j)}(\mathrm{DR}) = \text{const.}$; in other words, our algorithm drops the same number of copies for each content, regardless of its popularity. Finally, we describe CODA's operation and detail the steps necessary to implement CODA over the CCNx protocol [31].

3.1.1 Defining the Delivery-Rate Utility

We now define the delivery-rate utility ($U^{(i)}(\mathrm{DR})$) for a content i and devise a method to compute this value by estimating the request rate and the delivery rate of content i, using information that is locally available to each node. In our model, we assume that node inter-meeting times follow an exponential distribution with a constant parameter λ; in other words, if node a meets b, it expects to meet b again after $1/\lambda$ seconds elapse. Under this assumption, the probability that a receiver obtains a replica of content i is defined as follows:

Definition 1 *Given the node contact-rate λ, the average lifetime for an information request TTL, and the number of replicas $n^{(i)}$, the probability that a node obtains a copy of content i is equal to*

$$p^{(i)} = 1 - \exp\{-\lambda TTL \cdot n^{(i)}\} \ . \tag{3.1}$$

Hence, by duplicating content ($n^{(i)}$), increasing the average information-request lifetime (TTL), or augmenting the contact rate (λ), we improve the likelihood that a device obtains a copy of content i ($p^{(i)}$).

The delivery rate measures the number of replicas of a particular content i that are delivered to the intended destinations per unit of time:

Definition 2 *Given the request rate ($q^{(i)}$) and the probability of receiving a replica of content i ($p^{(i)}$), we define the delivery rate for content i as*

$$DR^{(i)} = q^{(i)} \cdot p^{(i)} \ . \tag{3.2}$$

Intuitively, as we increase the number of replicas of a content i, the probability that a node receives a copy of i increases and the delivery rate grows proportionally. CODA maximizes the total delivery rate, which is the sum of the delivery rates of each disseminated content i, subject to the following constraints: the total number of replicas of all content cannot exceed the amount of storage of all network nodes, a node cannot store duplicate copies of a content item i, and there must exist at least one replica of each content i in the network. This corresponds to the following optimization problem [24]:

$$\begin{aligned} \underset{n^{(1)},\ldots,n^{(k)}}{\text{maximize}} \quad & f(n^{(1)},\ldots,n^{(k)}) = \sum_{i=1}^{k} \mathrm{DR}^{(i)} \\ \text{subject to} \quad & \sum_{i=1}^{k} n^{(i)} - LB \leq 0, \\ & n^{(i)} - L \leq 0 \quad \text{for all content } i, \\ & n^{(i)} \geq 1 \quad \text{for all content } i, \end{aligned} \tag{3.3}$$

where k is the number of distinct content items, L is the number of nodes in the network, and B is the buffer capacity of each node.

Given that the constraints in (3.3) are linear [24] and assuming that $n^{(1)},\ldots,n^{(k)}$ are real random variables (and not integers), then this is a convex optimization problem, which can be solved efficiently [24]. CODA solves the above optimization problem in a distributed fashion by adopting a *gradient ascent* strategy [24], where the gradient of the objective function is the delivery-rate utility: $\nabla f = U(\mathrm{DR})$. In other words, the delivery-rate utility of content i is the marginal gain in the total delivery-rate with respect to the number of replicas of content i ($n^{(i)}$):

Definition 3 *Given the delivery rates for all k content types, $DR^{(1)},\ldots,DR^{(k)}$, the delivery-rate utility is defined as*

$$U^{(i)}(DR) = \frac{\partial}{\partial n^{(i)}} \sum_{j=1}^{k} DR^{(j)} \ . \tag{3.4}$$

Using the definition of the delivery rate (3.2), we can express $U^{(i)}(\mathrm{DR})$ in a more convenient form, as explained next.

Lemma 1 *The delivery-rate utility is proportional to the request-rate for content i ($q^{(i)}$) and decays exponentially with the number of replicas $n^{(i)}$:*

$$U^{(i)}(DR) = q^{(i)}(\lambda TTL)\exp\{-\lambda TTL \cdot n^{(i)}\} \ . \tag{3.5}$$

In other words, as nodes replicate a particular content i, the marginal gain in the delivery rate of i is gradually smaller; as $n^{(i)}$ tends to infinity, $U^{(i)}(\mathrm{DR})$ tends to zero.

Proof. Substituting (3.1) and (3.2) in (3.4) results in

$$\begin{aligned} U^{(i)}(\mathrm{DR}) &= \frac{\partial}{\partial n^{(i)}} \sum_{j=1}^{k} \mathrm{DR}^{(j)} \ , \\ &= \sum_{j=1}^{k} \frac{\partial}{\partial n^{(i)}} \left(\mathrm{DR}^{(j)}\right) \ , \\ &= q^{(i)} \frac{\partial p^{(i)}}{\partial n^{(i)}} \ , \\ &= q^{(i)}(\lambda \mathrm{TTL})\exp\{-\lambda \mathrm{TTL} \cdot n^{(i)}\} \ . \end{aligned} \tag{3.6}$$

Lemma 2 *The delivery-rate utility $U^{(i)}(DR)$ can, alternatively, be expressed as*

$$U^{(i)}(DR) = \lambda TTL(q^{(i)} - DR^{(i)}) \ . \tag{3.7}$$

Proof. We can re-write (3.6) in terms of the probability of receiving a replica of content i (3.1) as follows:

$$\begin{aligned} U^{(i)}(\mathrm{DR}) &= q^{(i)}(\lambda \mathrm{TTL}) \exp\{-\lambda \mathrm{TTL} \cdot n^{(i)}\} \ , \\ &= q^{(i)}(\lambda \mathrm{TTL})(1 - (1 - \exp\{-\lambda \mathrm{TTL} \cdot n^{(i)}\})) \ , \\ &= \lambda \mathrm{TTL}(q^{(i)} - q^{(i)} \cdot p^{(i)}) \ , \\ &= \lambda \mathrm{TTL}(q^{(i)} - \mathrm{DR}^{(i)}) \ . \end{aligned}$$

From (3.7), we know that the DR utility is proportional to the **miss rate**, defined as the number of unsatisfied requests for content i per second: $\mathrm{MR}^{(i)} = q^{(i)} - \mathrm{DR}^{(i)}$. CODA attempts to equalize this quantity for all content, irrespective of its request rate (or popularity). When a device's buffer is full, CODA discards the content replica with the lowest number of unsatisfied requests; by definition, this copy has the smallest impact on the total DR. If a node drops a content item, the network will have fewer replicas of this data, thus raising its DR utility (3.6). When a node meets a peer, it schedules the transmission of content according to the same utility function. In this case, the node chooses to send the content item with the greatest number of unsatisfied requests; by replicating this item, the device will increase the probability that a receiver obtains a copy of this content, thus reducing its utility (3.7). This gradient ascent strategy, illustrated in Figure 3.1, will converge to the number of replicas per content that maximizes the objective function f in (3.3), provided the iteration step is such that CODA converges to the optimal solution before the network changes substantially [24]. We now show that CODA will converge to a constant miss-rate per content by following the aforementioned strategy.

Theorem 1 *If nodes drop content by order of increasing delivery-rate utility, the following relationship will eventually hold:*

$$U^{(i)}(DR) = const. \qquad \text{for all content } i = 1, \ldots, k. \tag{3.8}$$

In other words, the proposed method drops the same number of replicas for each content type, regardless of its popularity.

Proof. Assume that all nodes have a full buffer. Let $\pi^- = \arg\min_i U^{(i)}(\mathrm{DR})$ and $\pi^+ = \arg\max_i U^{(i)}(\mathrm{DR})$. Node a requests content i, such that $U^{(\pi^+)}(\mathrm{DR}) \geq U^{(i)}(\mathrm{DR}) > U^{(\pi^-)}(\mathrm{DR})$. A node b that forwards i and that has not yet buffered i will discard the least useful content j in its buffer, where $U^{(i)}(\mathrm{DR}) > U^{(j)}(\mathrm{DR}) \geq U^{(\pi^-)}(\mathrm{DR})$. Then the delivery-rate utility of content j at time t can be expressed as follows:

$$\begin{aligned} U_t^{(j)}(\mathrm{DR}) &= q_{t-1}^{(j)} \exp\{-\lambda \mathrm{TTL}(n_{t-1}^{(j)} - 1)\} \ , \\ &= U_{t-1}^{(j)}(\mathrm{DR}) \exp\{\lambda \mathrm{TTL}\} \ . \end{aligned}$$

Analogously, the delivery-rate utility of content i will be updated as described next:

$$\begin{aligned} U_t^{(i)}(\mathrm{DR}) &= q_{t-1}^{(i)} \exp\{-\lambda \mathrm{TTL}(n_{t-1}^{(i)}+1)\} \ , \\ &= U_{t-1}^{(i)}(\mathrm{DR}) \exp\{-\lambda \mathrm{TTL}\} \ . \end{aligned}$$

Because $\lambda \mathrm{TTL} > 0$, $\exp\{\lambda \mathrm{TTL}\} > 1 > \exp\{-\lambda \mathrm{TTL}\}$ and the delivery-rate utility of i will decrease while that of j will increase.

If $j = \pi^-$ and $U_{t-1}^{(\pi^-)}(\mathrm{DR}) \exp\{\lambda \mathrm{TTL}\} \leq U_k^{(t)}(\mathrm{DR})$, for any content k, then

$$U_t^{(\pi^-)}(\mathrm{DR}) = U_{t-1}^{(\pi^-)}(\mathrm{DR}) \exp\{\lambda \mathrm{TTL}\} \ .$$

Analogously, if $i = \pi^+$ and $U_{t-1}^{(\pi^+)}(\mathrm{DR}) \exp\{-\lambda \mathrm{TTL}\} \geq U_k^{(t)}(\mathrm{DR})$, for any content k, then

$$U_t^{(\pi^+)}(\mathrm{DR}) = U_{t-1}^{(\pi^+)}(\mathrm{DR}) \exp\{-\lambda \mathrm{TTL}\} \ .$$

Eventually, we reach a time t^* where $U_{t^*}^{(\pi^+)}(\mathrm{DR}) = U_{t^*}^{(\pi^-)}(\mathrm{DR})$, at which time the following will hold

$$U^{(i)}(\mathrm{DR}) = \text{const.} \qquad \text{for all content } i = 1, \ldots, k.$$

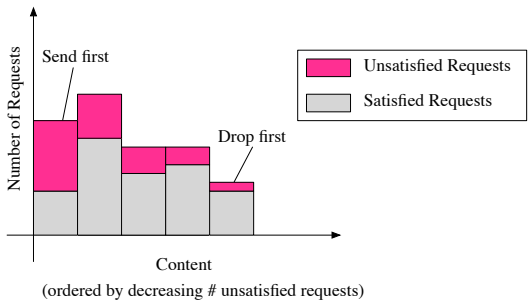


Fig. 3.1: CODA attempts to equalize the DR utility of all content. A device with a full buffer drops the content item with the lowest number of undelivered copies; discarding a content replica reduces the number of replicas in the network and increases the content's DR utility. When the same device meets a peer, it transmits the content item with the greatest number of unsatisfied requests; by increasing the number of content replicas, the probability that a receiver obtains a copy of this content rises, thus reducing the content's utility.

3.1.2 Estimating the Delivery-Rate Utility

To calculate the delivery-rate utility for a content i, a node must estimate the rate of request and delivery for i; it performs this estimate by recording the time when the application requests content i and when these requests are fulfilled. In order to improve this approximate value, a node merges its own rate estimates with those received from other network peers. Using the two approximate values for the request and delivery rate, the node computes the utility of content i by applying (3.7). Next, we define unbiased estimators for $q^{(i)}$ and $p^{(i)}$.

Definition 4 *Let $X_n^{(i)}$ be a RV denoting the n-th measurement of the number of requests for content i per node, issued by the application in the past W seconds, such that $\mathrm{E}[X_n^{(i)}] = q^{(i)}W$. Then our estimator for the request rate for content i, $q^{(i)}$, is given by*

$$\hat{q}_n^{(i)} = \alpha \hat{q}_{n-1}^{(i)} + (1-\alpha)\frac{X_n^{(i)}}{W} \ , \tag{3.9}$$

where $\hat{q}_n^{(i)}$ is the value of the estimator $\hat{q}^{(i)}$ after considering the n-th node meeting and $\alpha \in (0,1)$ is the weight given to the old estimate of the request-rate for content i, $\hat{q}_{n-1}^{(i)}$.

In other words, the estimator for the request rate of content i is computed as the exponentially weighted moving average (EWMA) of the observed samples $X_0^{(i)}$, $X_1^{(i)}, \ldots$, applying a weight of $(1-\alpha)$ to the most recent sample, $X_n^{(i)}$. We can show that the estimator obeys the following property:

Lemma 3 *The estimator for the request rate, $\hat{q}^{(i)}$, is unbiased:*

$$\mathrm{E}[\hat{q}^{(i)}] = q^{(i)} \ .$$

Proof. First note that $\hat{q}_0^{(i)} = X_0^{(i)}/W$, so $\mathrm{E}[\hat{q}_0^{(i)}] = \mathrm{E}[X_0^{(i)}/W] = q^{(i)}$. We can compute the expectation of $\hat{q}^{(i)}$ as follows:

$$\begin{aligned}
\mathrm{E}[\hat{q}^{(i)}] &= \mathrm{E}[\hat{q}_n^{(i)}] \ , \\
&= \mathrm{E}[\alpha \hat{q}_{n-1}^{(i)} + (1-\alpha)X_n^{(i)}/W] \ , \\
&= \alpha\,\mathrm{E}[\hat{q}_{n-1}^{(i)}] + (1-\alpha)\,\mathrm{E}[X_n^{(i)}/W] \ , \\
&= \alpha q^{(i)} + (1-\alpha)q^{(i)} \ , \\
&= q^{(i)} \ .
\end{aligned}$$

We now define the estimator for the delivery rate as follows.

Definition 5 *Let $Y_n^{(i)}$ be a RV denoting the n-th measurement of the number of satisfied requests for content i issued in the past t seconds, where $t \in [TTL, W+TTL]$, and such that $\mathrm{E}[Y_n^{(i)}] = p^{(i)}q^{(i)}W$. Then, our estimator for the delivery rate of content*

i, $\hat{DR}^{(i)}$, is given by

$$\hat{DR}_n^{(i)} = \beta \hat{DR}_{n-1}^{(i)} + (1-\beta) Y_n^{(i)}/W \ , \tag{3.10}$$

where $\hat{DR}_n^{(i)}$ is the value of the estimator after considering the n-th node meeting and $\beta \in (0,1)$ is the weight given to the previous estimate of $DR^{(i)}$.

In other words, the estimator for the delivery rate of content i is computed as the exponentially weighted moving average of the observed samples $Y_0^{(i)}, Y_1^{(i)}, \ldots$, applying a weight of $(1-\beta)$ to the most recent sample, $Y_n^{(i)}$. Here we shift back the time window by TTL seconds to allow the information requests issued in the last TTL seconds to receive an answer or, if none is received, to be classified as "unsatisfied". This estimator obeys the following property:

Lemma 4 *The estimator for the delivery rate, $\hat{DR}^{(i)}$, is unbiased:*

$$\mathrm{E}[\hat{DR}^{(i)}] = DR^{(i)} \ .$$

Proof. First observe that $\hat{\mathrm{DR}}_0^{(i)} = Y_0^{(i)}$, so $\mathrm{E}[\hat{\mathrm{DR}}_0^{(i)}] = \mathrm{E}[Y_0^{(i)}/W] = \mathrm{DR}^{(i)}$. We can now compute the expected value of $\hat{\mathrm{DR}}^{(i)}$ as described next:

$$\begin{aligned}
\mathrm{E}[\hat{\mathrm{DR}}^{(i)}] &= \mathrm{E}[\hat{\mathrm{DR}}_n^{(i)}] \ , \\
&= \mathrm{E}[\beta \hat{\mathrm{DR}}_{n-1}^{(i)} + (1-\beta) Y_n/W] \ , \\
&= \beta \, \mathrm{E}[\hat{\mathrm{DR}}_{n-1}^{(i)}] + (1-\beta) \, \mathrm{E}[Y_n^{(i)}/W] \ , \\
&= \beta \mathrm{DR}^{(i)} + (1-\beta) \mathrm{DR}^{(i)} \ , \\
&= \mathrm{DR}^{(i)} \ .
\end{aligned}$$

A mobile node couples its approximation of the request and delivery rate of a content i when it sends a replica of i to another device; this allows local estimates to permeate the network. CODA combines two estimates of the same rate by computing their arithmetic mean, which yields an unbiased estimate of the rate. An alternative approach consists in weighing each rate estimate by its inverse variance [7][22]. We analyze the properties of this combined estimator in Appendix A.

3.2 Content Optimal-Delivery Algorithm

The content optimal-delivery algorithm sits between the application and the network layer; it receives interest messages from the application and attempts to find matching content cached in the mobile device's buffer or available from a neighboring peer. When CODA fails to find matching content in the local buffer, it writes the query to the pending interests table of the CCNx protocol [31] until it can satisfy the

request. Periodically, each mobile device issues a **hello message** (HM) that signals its presence. When CODA captures a hello message it prepares a message with all its pending queries, known as the **pending interests message** (PIM); a mobile node within range attempts to satisfy the pending requests by searching its local storage and sending all matching content. CODA relies on a node's ability to estimate the request and delivery rate, $\hat{q}^{(i)}$ and $\hat{\mathrm{DR}}^{(i)}$, for a content in order to compute its delivery-rate utility, $U^{(i)}(\mathrm{DR})$. We assume that each content object i carries in its header the estimates for the two rates, which are calculated by the object's sender. When a mobile device's buffer is full, CODA uses the content utility to erase the least useful object from memory; similarly, when exchanging content with a neighbor, it prioritizes transmissions by decreasing utility. This insures that the most useful content reaches its neighbor in case both nodes part before the data-communication ends. Throughout this section, we analyze the aforementioned features in detail.

3.2.1 Processing Application Queries

When the application issues a content query, expressed as an interest message (IM), CODA records the time at which it received the interest (line 1 of Alg1) in the **statistics table** and searches the mobile device's buffer for matching content; the statistics table is a data structure, independent of the content buffer, that stores the time when requests are issued and when they are satisfied, from which CODA calculates the request and delivery rate of each content seen. If the buffer contains the desired content, CODA delivers this data immediately to the application, as shown in line 3 (Alg 1); otherwise, the device stores the query in the pending-interests table (line 6). The pending-interests table is a data structure that stores all the unsatisfied queries issued by the application; an interest remains in the PIT until CODA retrieves this data from a network peer. Periodically, CODA iterates through the statistics table to update the request and delivery rates of all content seen using equations (3.9) and (3.10).

Algorithm 1 Process interest messages from the application.

Input: interest for content i from application
1: **record** time of request for i in statistics table
2: **if** buffer contains i **then**
3: **deliver** content i to application
4: **record** time when request for i satisfied in statistics table
5: **else**
6: **write** interest to pending-interests table
7: **end if**

A mobile device cannot satisfy all content requests with its cache alone; in such cases, the device stores the query in the PIT while waiting for a response. Peri-

odically, a device emits a hello message, which serves to alert other nodes of its presence; the delay for transmitting the next HM, d_{hello}, is calculated as follows:

$$d_{\text{hello}} = d_{\text{hello}}^{\min}\left(b_{\text{hello}} + r\right) \quad , \tag{3.11}$$

where $d_{\text{hello}}^{\min}$ is the minimum delay between HM messages, $b_{\text{hello}} \in [0, b_{\text{hello}}^{\max}]$ is the hello backoff-counter, $b_{\text{hello}}^{\max}$ is the maximum hello-backoff, and $r \in [0,1]$ is a random value. By randomizing the transmission delay of the HM, CODA reduces the probability of interfering with ongoing but yet undetected transmissions. When a mobile device receives a hello message, it delays the transmission of its own periodic message by increasing the backoff counter (b_{hello}), as shown in lines 1–3 of Alg. 2; this avoids congesting the network with an excessive amount of hello packets. As a response to the hello message, CODA prepares a pending-interests message, comprising the application's unsatisfied requests, and schedules the transmission of the message to all device's within range (lines 4–8 of Alg. 2).

Algorithm 2 Processing of hello messages.

Input: hello message from network
1: **if** a hello message is waiting to be sent **then**
2: **delay** transmission of hello message
3: **end if**
4: **create** new pending interests message
5: **for** each interest in the pending interests table **do**
6: **write** interest to pending interests message
7: **end for**
8: **schedule** transmission of pending interests message

Our algorithm computes the transmission delay of the PIM, d_{PIM}, as follows:

$$d_{\text{PIM}} = d_{\text{PIM}}^{\min}\left(b_{\text{PIM}} + r\right) \quad , \tag{3.12}$$

where $d_{\text{PIM}}^{\min}$ is the minimum waiting time before transmitting the PIM, $b_{\text{PIM}} \in [0, b_{\text{PIM}}^{\max}]$ is the PIM backoff-counter, bounded by 0 and the maximum backoff ($b_{\text{PIM}}^{\max}$), and $r \in [0,1]$ is a random value. When a node receives a PIM, it defers the transmission of a (possibly) pending PIM by increasing the backoff counter, b_{PIM} (lines 1–3 of Alg. 3); this enables a mobile device to satisfy part or all of its own pending interests by overhearing the upcoming data-exchange, thus reducing the number of entries in its next PIM. To process an incoming PIM, a node retrieves from its buffer all the content items that satisfy the interests expressed in the PIM (lines 4–8 of Alg. 3); it then orders the sought content by decreasing DR utility and couples the estimated request and delivery rates to each content object sent (lines 9–13 of Alg. 3).

Algorithm 3 Processing of pending-interests messages.

Input: pending interest message from network
1: **if** a pending interests message is waiting to be sent **then**
2: **delay** transmission of pending interests message
3: **end if**
4: **for** each interest in the pending-interests message **do**
5: **if** exists matching content *i* in buffer **then**
6: **write** content *i* to list
7: **end if**
8: **end for**
9: **sort** list in order of decreasing DR utility
10: **for** each content *i* in the list **do**
11: **write** request and delivery rate to content *i*
12: **schedule** transmission of content *i*
13: **end for**

3.2.2 Processing Incoming Content

Whenever a mobile device receives a content object, it checks the PIT for interests corresponding to this data; in case the content object matches an interest in the PIT, CODA delivers the data to the application and records the time at which the interest was satisfied (lines 1–4 of Alg. 4). As each mobile device may have its own estimate of the request and delivery rates, CODA computes the arithmetic mean of the two estimates and updates the statistics table, as shown in lines 5–6 (we discuss the properties of this combined estimate in Appendix A). Finally, CODA caches the replica of content *i* if no such copy exists in the device's buffer (line 10).

Algorithm 4 Process incoming content from the network.

Input: datagram with content *i* from network
1: **if** exists pending interest for content *i* **then**
2: **deliver** content *i* to application
3: **record** time when request for *i* satisfied in statistics table
4: **end if**
5: **average** request-rate in statistics table and rate in datagram (A.3)
6: **average** delivery-rate in statistics table and rate in datagram (A.3)
7: **if** buffer contains *i* **then**
8: **discard** content *i*
9: **else**
10: STORE-CONTENT(*i*)
11: **end if**

Function STORE-CONTENT implements CODA's buffer management strategy (Alg. 5). Our algorithm accepts to store all unique content replicas published by the user and those received from the network as long as there is enough storage capacity (lines 2–5). Whenever the device's buffer is full, CODA selects the replica with the

lowest delivery-rate utility from memory, say m, and compares it with the utility of replica i; CODA keeps only the replica with the highest utility from the pair $\{i,m\}$ (lines 6–12). By definition, the replica with the lowest utility has the least impact on the total content delivery-rate, if dropped.

Algorithm 5 Store content in the buffer.

```
1:  function STORE-CONTENT(i)
2:      if buffer not full then
3:          write content i to buffer
4:          return true
5:      end if
6:      let m ← content with lowest DR utility in buffer
7:      if i's utility > m's utility then
8:          replace replica m with i in buffer
9:          return true
10:     end if
11:     discard content i
12:     return false
13: end function
```

3.2.3 Implementation of CODA

In this section we describe the necessary steps to implement CODA over the latest revision of the CCNx protocol[1] [31]. The CCNx protocol implementation is divided into two main components: the client, which acts as a mediator between the server and the application and provides an interface for the application to issue data requests and obtain content objects from the network, and the server, which propagates data requests to the network and obtains content objects from other servers. The principal changes required to deploy CODA over CCNx consist in modifying the header of the content object to include the request and delivery-rate estimates, disabling the propagation of interest messages to network peers, and introducing two control packets that implement the hello and pending-interests messages.

A content object contains four units: `Signature`, `Name`, `SignedInfo`, and `Content`, as illustrated in Figure 3.2. The `Signature` contains a digest of the `Name`, `SignedInfo`, and `Content` components, computed using a signature algorithm [31]; the `Name` unit holds the hierarchical CCNx name of the content; the `SignedInfo` component specifies the signer of the message, the time instant when the object was signed, the type of content conveyed, and the message's expiration time. The original CCNx server stores each content object received from the network in a hash table. For each content replica received, the server divides the content object into two parts: the first part comprises the `Signature` and `Name` units and the second consists of the remaining components of the content object. The

[1] At the time of this writing, the latest revision of the CCNx protocol was 0.6.0.

server computes a hash value of the first part of the content object to index each replica stored in the buffer. We introduced the estimates of the request and delivery rates as two integer fields in the `SignedInfo` section of the message. Because the `SignedInfo` section belongs to the second half of the content object, recomputing the two rate estimates for each stored replica does not affect its hash value; hence, the server can still locate content in the buffer after updating the rate estimates of the stored replicas.

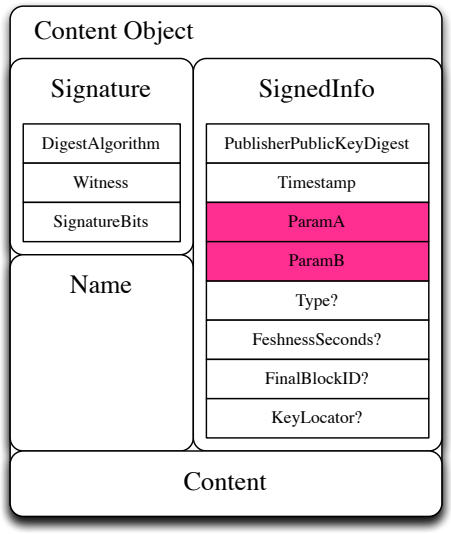


Fig. 3.2: The structure of the CCNx content object [31][19]. A content object comprises four main compartments: signature, name, signed info, and content. A host generates the digital signature based on the binary encoding of the name, signed info, and content sections of the content object. The signed info component contains data that helps a CCNx device verify a digital signature. The name section identifies the data stored in the object; it comprises a sequence of components, where each component contains a succession of zero or more bytes (e.g. `/epfl.ch/IC/videos/welcome.mpg`). Lastly, the content portion carries the actual data. Each field marked with a "?" sign is optional. We added the two fields marked in **red** to the content object, param-*A* and param-*B*, that carry the request- and delivery-rate, respectively.

A message in the CCNx protocol is encoded using the **CCNx binary** (CCNB) format [31]. Each section of a message is labeled by a `DTAG`, which is a unique identifier drawn from an internal dictionary. The parser or decoder outputs the byte number at which each message element starts and ends. We encode each estimate as a four-byte `BLOB`, representing an integer value, enclose each estimate with a

DTAG, and modify the content-object parser so as to locate the boundaries of the two values. By using BLOBs of fixed length, we can recompute the estimates for the two rates, encode them using the CCNB format, and replace the old values in the content object without generating a new message-header. Figure 3.2 illustrates the components of the content object, including the section where we encode the two estimated rates.

To compute the value of the two rates, we added a statistics table to the server to record the history of content requests; by storing content-requests as a separate data structure we ensure that its lifespan is independent of other server events, such as the removal of stale content from the buffer. Each table entry contains a list of all requests issued and satisfied in the preceding $W + \mathrm{TTL}$ seconds for a specific content, where W denotes the sampling period and TTL is the average lifetime of an information request; each entry is indexed by the CCNx name of the content being tracked. To calculate the value of the request-rate estimator, $\hat{q}^{(i)}$, we apply (3.9) and measure $X_n^{(i)}$ by counting the number of requests sent in the preceding W seconds recorded in the content-requests table. We estimate the delivery rate, $\hat{\mathrm{DR}}^{(i)}$, using (3.2), and evaluate $Y_n^{(i)}$ by tallying all the satisfied requests transmitted in the preceding $[\mathrm{TTL}, \mathrm{W} + \mathrm{TTL}]$ seconds. Knowing the two estimates, $\hat{q}^{(i)}$ and $\hat{\mathrm{DR}}^{(i)}$, we determine the utility of content i using (3.7) and record this value alongside each replica stored in the buffer. Whenever the buffer exceeds a storage threshold, which is controlled by the parameter CCND_CAP [31], we sort the buffer in order of increasing utility and discard the least-useful replicas until the buffer size is below the storage limit.

We implement CODA's hello and pending-interests messages as content objects. The hello message consists of a content object with Name set to /coda/hello, type field set to CCN_CONTENT_HELLO, and having an empty Content section. A pending-interests message has a CCNx name of /coda/pit, type field set to CCN_CONTENT_PIT, and the Content section of the message holds a list of the content-names requested by the application and stored in the source node's pending-interests table; this message is generated as a response to a hello message. Upon receiving a pending-interests message, CODA reads each name entry in the PIM's payload and locates matching content in the device's memory; the algorithm sorts all matching content by decreasing utility before scheduling each transmission. We use CCNx's inbuilt scheduler to implement the transmission delay equations (3.11) and (3.12) for the PIM and HM, respectively. Additionally, we rely on CCNx's procedure for sending data that introduces an additional random delay before each transmission; this helps reduce the likelihood of colliding with ongoing communications. The CCNx protocol's forwarding-information base comprises a list of IP addresses for dispatching interest messages for which there is no matching content in the local content-store. Because we use the hello and pending-interests messages for a similar purpose, we disabled the interest-forwarding procedure found in CCNx. However, we use the FIB table to store a default entry with CCNx name / that points to CODA's multicast address; a device always uses this address to send data to its peers over the network.

Chapter 4
Evaluation

Abstract We want to confirm if CODA maximizes the throughput of the network by equalizing the delivery-rate utility of all disseminated content. Additionally, we want to measure the effect of the buffer capacity per node, network load, and the distribution of content popularity on the network throughput. For this purpose we deploy CODA in a simulation network comprising 50 nodes that move according to a random-direction model. Nodes moving according to this model have exponentially distributed or, at least, exponentially tailed inter-meeting times, which agrees with our initial design assumptions. We simulate our proposal using network simulator 3 (NS3) and use the direct-code execution (DCE) feature to deploy the same version of CODA conceived for real mobile-devices on the simulation nodes. Although this simulation method posed a significant performance penalty, our results gained credence. According to the outcome of the simulation, CODA achieves a constant DR utility for all content, given enough time and network resources. The congestion avoidance mechanism effectively scales down data traffic when the network is overloaded. Increasing the buffer capacity per node raises the network throughput as long as this capacity does not exceed the total number of disseminated contents. By raising the Zipf exponent, nodes request only a fraction of the available contents and can satisfy most user-requests with less buffer space, which justifies the proportional increase in the network throughput.

4.1 Simulation of CODA

The purpose of simulating CODA is to verify if it maximizes the total delivery-rate by equalizing the DR utility of all content; hence, for a sufficient number of content requests, the DR utility should converge to a constant, as claimed by Thm 1. To test our proposal, we deploy CODA in a network of 50 nodes, which move according to a random-direction model, travel at a constant speed of 50 Km/h within a square field of width 500m, and change direction every two seconds. The authors in [24] show that nodes moving according to a random-direction model have exponentially-

distributed or, at least, exponentially-tailed inter-meeting times; hence, the chosen mobility model is appropriate to validate our proposal. Additionally, each node publishes exactly one content item at the beginning of the test and requests 5 content items published by other nodes every 30 seconds plus a random interval of 5 seconds; this random interval stops nodes from issuing synchronous queries, which would generate an unrealistic volume of network traffic. We assume that content does not expire and that there is, at least, one replica of each content in the network. We model the popularity of content according to a Zipf distribution; hence, each node has a probability of requesting a specific content that is defined according to this distribution. Finally, all mobile devices communicate using IEEE 802.11 radios with a data rate of 1 Mbps and have a transmission range of 100m. For each content, we collect the number of requests and the number of satisfied requests and average the data gathered over three (independent) trials. Using this network scenario, we verify if, indeed, CODA achieves a constant DR utility for all content, irrespective of its popularity, and we measure the influence of the network load, buffer size, and the exponent of the Zipf distribution on this constant. Table 4.1 summarizes the unchangeable attributes of our network scenario.

Parameter	Value
Network size	50 nodes
Deployment field	500 m $\times$ 500 m
Mobility model	random-direction
Node speed	50 Km/h (constant)
Node travel direction	changes every 2 sec
Content items published per node	1 content item
Periodicity of content requests	5 contents every 30 to 35 sec
Node radios	IEEE 802.11 with a data-rate of 1 Mbps
Transmission range	100m
Data collected	number of requests per content number of satisfied requests per content
Number of Trials	3 (independent) test runs

Table 4.1: The table lists the value of each simulation parameter we kept fixed throughout the tests performed on CODA.

Each node runs an instance of the CCNx server process [31], `ccnd`, which contains the implementation of CODA. We model the behavior of the user's application by generating events from within the simulation script. Initially, each application calls the `ccndc` process [31] to create a default entry in the forwarding-information base, named "/" (forward slash), that points to the multicast address, 225.1.2.4, and multicast port, 9695, through which all inter-device communication takes place. Because content does not expire, each application publishes content objects once, at the beginning of the simulation, by executing the `ccnpoke` command [31], and stores

these objects permanently in the node's buffer[1]. The application requests content by executing the `ccnpeek` command [31], which generates an interest message with the name of the desired content. The `ccnpoke`, `ccnpeek`, `ccndc`, and `ccnd` processes interchange data using Unix sockets. All nodes transmit data using the UDP transport protocol, running over an IP network. Figure 4.1 illustrates the protocol stack of our simulation nodes.

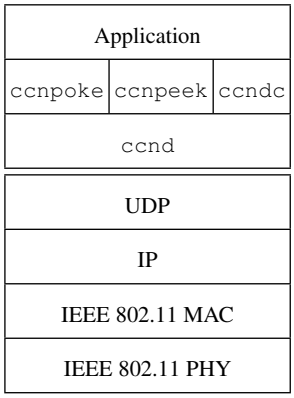


Fig. 4.1: Protocol stack of the simulation nodes. We model the behavior of the user's application by generating events from the simulation script. The application publishes content by executing the `ccnpoke` process, generates interest messages by executing `ccnpeek`, and creates a default entry in the forwarding-information base for transmitting multicast datagrams using the `ccndc` command. The `ccnpoke`, `ccnpeek`, `ccndc`, and `ccnd` processes interchange data using Unix sockets. Each device transmits data using the UDP protocol, running over an IP network.

To simulate our algorithm and setup the aforementioned network scenario, we use network simulator 3 (NS3) [13]. The direct code-execution (DCE) feature of this simulator [26] enabled us to run CODA, implemented over CCNx, from within the simulation, without requiring further changes to our source code. However, the DCE component also introduced a significant performance penalty; a simulation lasting 2000 seconds for a network of 50 nodes can take longer than 30 minutes to complete on a laptop powered by the latest quad-core Intel Core i7 processor. Additionally, each time our code exhibited a fault and called the `abort()` instruction [23], the abort signal is caught by the DCE component and the execution of the other simulation threads continues, thus masking important errors in the code. DCE generates four files for each process launched from the simulation; as there is no procedure to disable this, we end up with an explosion of useless files at the end of each simulation. Despite these drawbacks, we adopted this simulation strategy as

[1] We tag such content items as "precious" [31] to thwart any attempt from the `ccnd` process to remove this data from storage.

by deploying the same version of CODA conceived for real mobile-devices on the simulation nodes, we gave credibility to our results.

4.2 Analysis of Results

We tested CODA according to the parameters described in Table 4.1, for a total simulation time of $T = 4000$ seconds. In this test, the popularity of content follows a Zipf distribution with parameter $\gamma = 2/3$; we set γ to the same value proposed in [15], computed by analyzing the popularity of 8000 content feeds, accessed by more than 1 million users. Additionally, each node requests $R = 5$ content items every 30–35 secs and has a buffer capacity of $B = 11$; we reserve 1 slot for the content item published by the application, 5 slots for storing content retrieved from peer devices, and the remaining 5 slots for caching CCNx's control data. Because we claim that our algorithm maximizes the total delivery-rate (3.3), we must verify if the simulation results evidence a constant miss rate for all disseminated content, irrespective of its popularity.

Recall that the utility of content i is defined as $U^{(i)}(\mathrm{DR}) = \lambda \mathrm{TTL} \cdot \mathrm{MR}^{(i)}$, where $\mathrm{MR}^{(i)} = q^{(i)} - \mathrm{DR}^{(i)}$ is i's miss rate. Assume that the optimization problem (3.3) has only one constraint:

$$\begin{aligned} \underset{n^{(1)},\ldots,n^{(k)}}{\text{maximize}} \quad & f(n^{(1)},\ldots,n^{(k)}) = \sum_{i=1}^{k} \mathrm{DR}^{(i)} \\ \text{subject to} \quad & \sum_{i=1}^{k} n^{(i)} - LB \leq 0. \end{aligned}$$

Further, let $g(n^{(1)},\ldots,n^{(k)}) = \sum_{i=1}^{k} n^{(i)} - LB$. Using Lagrange multipliers, we want to find a point $(n^{(1)},\ldots,n^{(k)})$ where f is maximal, which occurs when $\nabla f = \varsigma \nabla g$ and $g(n^{(1)},\ldots,n^{(k)}) \leq 0$ hold, where ς is the Lagrange multiplier. From Definition 3 we know that

$$\nabla f = (U^{(1)}(\mathrm{DR}),\ldots,U^{(k)}(\mathrm{DR})) \tag{4.1}$$

and the gradient of g is given by

$$\nabla g = (1,\ldots,1) \ , \tag{4.2}$$

a vector of k 1's. Re-arranging (4.1) and (4.2) with respect to the Lagrange multiplier gives

$$U^{(i)}(\mathrm{DR}) = \varsigma \qquad \text{for all content } i. \tag{4.3}$$

This is why we expect to obtain a constant utility or miss rate for all content at the point $(n^{(1)},\ldots,n^{(k)})$ where f is maximal. Additionally, we know from Thm 1 that CODA achieves a constant utility upon convergence.

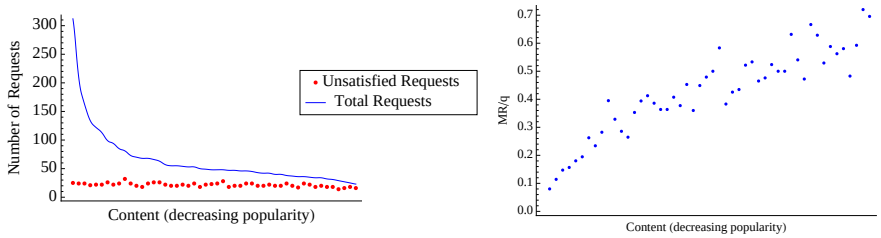
Figure 4.2a shows a plot of the total number of requests (blue line) and the number of unsatisfied requests or miss rate (red dots) for each content, listed in order of decreasing popularity. It is clear from this plot that there is no correlation between the popularity of content and the number of unsatisfied requests per content, which corroborates Thm 1. Consequently, we expect content with low popularity to suffer huge losses, measured as the number of unsatisfied requests over the total number of requests issued; this result is shown in Figure 4.2b. To achieve fairness, at the cost of throughput, we must define the objective function f in (3.3) as

$$f(n^{(1)},\ldots,n^{(k)}) = \sum_{i=1}^{k} \frac{\mathrm{DR}^{(i)}}{q^{(i)}} \quad , \tag{4.4}$$

and derive the utility function for each content i as

$$U^{(i)}(\mathrm{DR/q}) = \frac{\partial}{\partial n^{(i)}} f(n^{(1)},\ldots,n^{(k)}) \quad , \tag{4.5}$$

in which case we would expect the plot in 4.2b to exhibit a constant line, implying that the fraction of unsatisfied requests over the number of requests would be constant for all content.



(a) Constant MR (b) Increasing MR/q

Fig. 4.2: We simulated CODA for a network with $L = 50$ nodes for a total simulation time of $T = 4000$ secs, where the popularity of content follows a Zipf distribution with parameter $\gamma = 2/3$. Each node has a buffer capacity of $B = 11$ slots (1 slot = 1 content) and requests $R = 5$ content items every 30–35 secs. CODA maximizes the total delivery-rate by equalizing the number of unsatisfied requests for all content disseminated in the network. As expected, the plot in Figure (4.2a) evidences a constant number of unsatisfied requests for all content, regardless of its popularity. CODA does not attempt to equalize the MR$/q$ value, hence the proportion of unsatisfied requests per number of requests increases as the popularity of content decreases; this phenomenon is illustrated in Figure (4.2b).

We conduct three tests, lasting $T = 2000$ seconds each, to measure CODA's normalized throughput for varying buffer capacities per node, different network loads,

and distinct exponents for the Zipf distribution of content popularity; the **normalized throughput** is the total number of satisfied requests divided by the total number of requests, for all nodes and content. In the first test we increase the buffer capacity per node (B) from 5 to 30 in steps of 5; in the second test we decrease the minimum delay between hello messages from 5 seconds to 0.5 seconds in steps of -0.5 seconds ($d_{\text{hello}}^{\min}$); and in the third test, we increase the exponent for the Zipf distribution of content popularity (γ) from 0 to 2 in steps of $1/3$. The parameters of the three tests are summarized in Table 4.2.

Parameter	Test 1	Test 2	Test 3
Buffer size (# items stored) (B)	from 5 to 30, steps of $+5$	fixed 11	fixed 11
Hello message delay (s) ($d_{\text{hello}}^{\min}$)	fixed 5	from 5.0 to 0.5, steps of -0.5	fixed 5
Zipf exponent (γ)	fixed $2/3$	fixed $2/3$	from 0 to 2, steps of $+1/3$
Measurement	Total # of satisfied requests over the total # of requests		

Table 4.2: We measure CODA's normalized throughput for varying buffer capacities per node, different network loads, and distinct exponents for the Zipf distribution of content popularity. In test 1 we vary the buffer capacity per node (B); in test 2 we vary the minimum delay between hello messages ($d_{\text{hello}}^{\min}$); and in test 3 we vary the exponent of the Zipf distribution for the popularity of content (γ). In all tests we measure the total number of satisfied requests divided by the total number of requests, for all content and for all nodes.

As we grow the buffer capacity per node, we expect the normalized throughput achieved by CODA to increase. This is because the total DR increases with $n^{(1)},\ldots,n^{(k)}$ and the first constraint in (3.3), $\sum_i n^{(i)} - LB \leq 0$, states that we may replicate content as long as the network can store this data; the network capacity is given by LB, where L is the number of nodes and B is the storage capacity per node. We can observe this behavior in the plot of the normalized throughput for test 1 shown in Figure 4.3; the normalized throughput increases linearly as we grow the buffer size from 5 to 30 items. Because of the constraint $\forall_i : n^{(i)} - L \leq 0$ (3.3), nodes cannot store duplicates in their inventories; thus, increasing the cache size per node beyond the amount of distinct contents in the network will not improve the normalized throughput further, as shown in Figure 4.3.

To measure the impact of the network load on the normalized throughput, we vary the minimum delay between consecutive hello messages ($d_{\text{hello}}^{\min}$); recall that a hello message initiates a conversation between two mobile devices. Hence, by reducing $d_{\text{hello}}^{\min}$, we increase the network traffic. The plot in Figure 4.4 evidences a

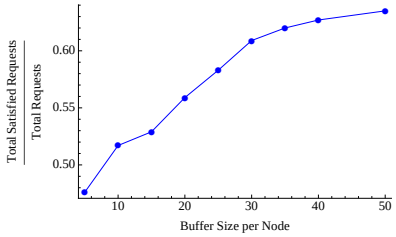


Fig. 4.3: Influence of the buffer capacity on CODA's normalized throughput. Because the objective function in (3.3) is an increasing function of the number of replicas, we can improve the normalized throughput by replicating content as long as the nodes can store this data. Given that nodes cannot cache duplicate contents, expanding the buffer size beyond the number of distinct contents in the network will not improve the normalized throughput further.

linear relationship between $d_{\text{hello}}^{\min}$ and the normalized throughput (blue line):

$$\text{NTP} = a d_{\text{hello}}^{\min} + b \ , \tag{4.6}$$

where $a = 8.80 \times 10^{-3}$ and $b = 5.85 \times 10^{-1}$. The network is severely congested when we decrease $d_{\text{hello}}^{\min}$ from 2.5 seconds to 1 second (red line); here, we observe a power-law relationship between $d_{\text{hello}}^{\min}$ and the normalized throughput:

$$\text{NTP} = a (d_{\text{hello}}^{\min})^b + c \ , \tag{4.7}$$

where $a = 4.63 \times 10^{-4}$, $b = 5.13$, and $c = 5.57 \times 10^{-1}$. When $d_{\text{hello}}^{\min}$ approaches

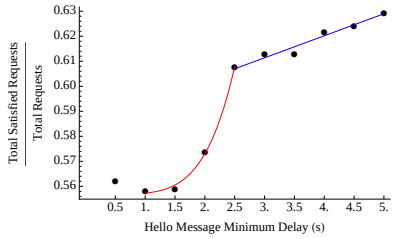


Fig. 4.4: Influence of the network load on CODA's normalized throughput.

1 second, the normalized throughput no longer decreases. This is due to CODA's congestion-avoidance mechanism: when a mobile device hears a hello message, it differs the transmission of its own hello packet to avoid congesting the network.

We assume that the popularity of content in the network follows a Zipf distribution. Hence, the probability that a node requests a given content $i \in [1,k]$ is given by

the distribution's PDF:

$$\Pr\{\text{a node requests } i\} = \frac{i^{-\gamma}}{\sum_{j=1}^{k} j^{-\gamma}} \quad , \tag{4.8}$$

where γ is the only tunable parameter of this distribution. As we increase γ, nodes will request only a small fraction of the available content. Because nodes do not store duplicates, each device will be able to satisfy most content requests with only a fraction of its available memory. Hence, we expect an increase in the normalized throughput as we raise γ; this phenomenon is illustrated in Figure 4.5. In this plot, we can also observe a low throughput when γ is zero. This is because all 50 content items have an equal probability of being picked, i.e. $1/k$, and each device can store only 10% of the available content[2].

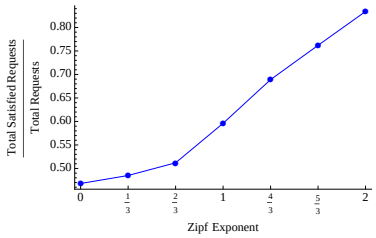


Fig. 4.5: Influence of the distribution of content popularity on CODA's normalized throughput. As we increase the Zipf exponent (γ), nodes request only a small fraction of the available content and require less storage to cache the most popular content; because each device can satisfy more queries using less space, the normalized throughput increases. However, when $\gamma = 0$, contents have equal probability of being picked; because for $B = 11$ each device can store only 10% of the available content, the normalized throughput is lower.

[2] Although the buffer capacity per node is $B = 11$, each device uses only ≈ 5 slots for storing requested content.

Chapter 5
Conclusion

We designed the content optimal-delivery algorithm for disseminating named data over a delay-tolerant network. A delay-tolerant network interconnects challenged networks, whose links are characterized by high latency, low and (possibly) asymmetric data-rates, and frequent disconnections. We assume that a user employing CODA carries a mobile device, such as a smartphone, that can connect directly to other similar devices over wireless links. When a user requests a specific content, the algorithm searches the device's cache for matching data; if the content query cannot be satisfied locally, CODA stores the request in a table. Periodically, each mobile device issues a hello message, which triggers a conversation between nearby devices. A wireless node receiving a hello message broadcasts a pending-interests message with a list of user queries that could not be satisfied locally. Nearby devices attempt to satisfy this query list by ferreting their own content storage. When a mobile device receives content that satisfies a pending request, it delivers this data to the user.

CODA maximizes an objective function, defined as the the total number of satisfied requests or network throughput, subject to the following constraints: (i) the number of content replicas cannot exceed the total storage capacity of the network, which is equal to the product of the number of network devices and the buffer capacity per node, (ii) a device cannot store duplicate items in its cache, and (iii) the publisher of a content must store this replica permanently in its buffer. For this purpose, we derive the per-content DR utility as the gradient of the objective function; CODA estimates this value using network data that is available locally to each node. The utility function quantifies the impact of replicating and dropping a content item on the system's total delivery rate; it is proportional to the content miss-rate or the number of unsatisfied content-requests per unit of time. We show that the delivery rate is maximal when the DR utility of all content equals a constant; hence, CODA must strive to equalize the DR utility of all content in order to optimize the network throughput. When a device's buffer is full, our algorithm discards the least useful content first; additionally, when a mobile device transmits a batch of data to a peer, it sends the most useful content first. The simulation results suggest that by following

this strategy CODA achieves a constant miss rate, given enough network resources and time.

We engineered CODA with a congestion-avoidance mechanism to reduce the risk of packet loss in a wireless medium and avoid the cost of retransmission in a challenged network. When a pair of devices initiates a conversation, CODA increases the period between hello messages to cut back network traffic between nearby nodes. Similarly, upon receiving a pending-interests message, a CODA device postpones the transmission of its own PIM; this strategy allows nodes to get content they are interested in just by listening to the medium. Furthermore, CODA always waits a short, random delay before transmitting a content object to mitigate the risk of interfering with ongoing and yet undetected communications. The simulation results evidence the success of the congestion aversion measures: decreasing the minimum delay between hello messages when the network is jammed does not further reduce the overall throughput.

Although the initial results are promising, we must perform additional tests with bigger sized networks, comprising hundreds of nodes that publish and subscribe a greater amount of content, and measure the convergence time of the utility estimators. We need to compare CODA's "optimal" delivery rate with the maximal delivery rate obtained by solving the constrained optimization problem in (3.3). Additionally, we must validate the simulation results by deploying CODA on real mobile devices before concluding that the algorithm is suitable for distributing named content in delay-tolerant networks. As an extension of this work, we plan to experiment with different utility functions in order to (i) achieve equal throughput for all content, independent of its popularity, and (ii) minimize the content delivery-delay.

References

1. Balakrishnan, H.: Wireless Channel Access Protocols. Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA (2005). URL http://nms.csail.mit.edu/6.829-f05/lectures/L11-wlessmac.pdf
2. Balasubramanian, A., Levine, B., Venkataramani, A.: DTN routing as a resource allocation problem. ACM SIGCOMM Computer Communication Review **37**(4), 373 (2007). DOI 10.1145/1282427.1282422. URL http://portal.acm.org/citation.cfm?doid=1282427.1282422
3. Bharghavan, V., Demers, A., Shenker, S., Zhang, L.: MACAW: a media access protocol for wireless LAN's. In: Proceedings of the conference on Communications architectures protocols and applications, vol. 24, pp. 212–225. ACM (1994). DOI 10.1145/190314.190334. URL http://portal.acm.org/citation.cfm?id=190334
4. Boldrini, C., Conti, M., Passarella, A.: Design and performance evaluation of ContentPlace, a social-aware data dissemination system for opportunistic networks. Computer Networks **54**(4), 589–604 (2010). DOI 10.1016/j.comnet.2009.09.001. URL http://linkinghub.elsevier.com/retrieve/pii/S1389128609002783
5. Broder, A., Mitzenmacher, M.: Network Applications of Bloom Filters: A Survey. Internet Mathematics **1**(4), 485–509 (2004). DOI 10.1.1.127.9672. URL http://akpeters.metapress.com/index/V7324576U920P665.pdf
6. Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., Weiss, H.: Delay-Tolerant Networking Architecture (2007). URL http://www.ietf.org/rfc/rfc4838.txt
7. Duffield, N., Lund, C., Thorup, M.: Optimal Combination of Sampled Network Measurements. Internet Measurement Conference pp. 91—-104 (2005). URL http://www2.research.att.com/ duffield/papers/DLT05-combine.pdf
8. Fall, K.: A delay-tolerant network architecture for challenged internets. Proceedings of the 2003 conference on Applications technologies architectures and protocols for computer communications SIGCOMM 03 **25**, 27 (2003). DOI 10.1145/863955.863960. URL http://portal.acm.org/citation.cfm?doid=863955.863960
9. Gritter, M., Cheriton, D.R.: An architecture for content routing support in the internet. Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and SystemsVolume 3 **15**(5), 4 (2001). DOI 10.1145/1570499.1570501. URL http://portal.acm.org/citation.cfm?id=1251444
10. Han, B., Hui, P., Kumar, V.S.A., Marathe, M.V., Pei, G., Srinivasan, A.: Cellular Traffic Offloading through Opportunistic Communications : A Case Study. System pp. 31–38 (2010). DOI 10.1145/1859934.1859943. URL http://dl.acm.org/citation.cfm?id=1859943
11. Hart, D.: A Brief History of NSF and the Internet. Tech. rep., National Science Foundation, Arlington, Virginia, USA (2003). URL http://www.nsf.gov/od/lpa/news/03/fsnsf_internet.htm
12. Helgason, O.R., Yavuz, E.A., Kouyoumdjieva, S.T., Pajevic, L., Karlsson, G.: A Mobile Peer-to-Peer System for Opportunistic Content-Centric Networking. Transport pp. 21–26 (2010). DOI 10.1145/1851322.1851330. URL http://portal.acm.org/citation.cfm?id=1851330
13. Henderson, T.R., Riley, G.F.: Network Simulations with the ns-3 Simulator. Computer Engineering pp. 2006–2006 (2006). URL http://conferences.sigcomm.org/sigcomm/2008/papers/p527-hendersonA.pdf
14. Hiertz, G., Denteneer, D., Stibor, L., Zang, Y., Costa, X.P., Walke, B.: The IEEE 802.11 universe (2010). DOI 10.1109/MCOM.2010.5394032. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5394032
15. Hu, L., Le Boudec, J.Y., Vojnoviae, M.: Optimal Channel Choice for Collaborative Ad-Hoc Dissemination. 2010 Proceedings IEEE INFOCOM pp. 1–9 (2010). DOI 10.1109/INFCOM.2010.5462163. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5462163
16. Hui, P., Crowcroft, J., Yoneki, E.: Bubble rap: social-based forwarding in delay tolerant networks. IEEE Transactions on Mobile Computing **10**(November), 241–250 (2008). DOI 10.1145/1374618.1374652. URL http://portal.acm.org/citation.cfm?id=1374618.1374652

17. IEEE 802.11 Standard-1997: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Tech. rep., IEEE (1997)
18. International Telecommunication Union: Active mobile-broadband subscriptions per 100 inhabitants. Tech. rep. (2011). URL http://www.itu.int/ITU-D/ict/statistics/material/excel/20112/ictwebsite/Mobile_bb_11.xls
19. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. Proceedings of the 5th international conference on Emerging networking experiments and technologies CoNEXT 09 **178**(1), 1–12 (2009). DOI 10.1145/1658939.1658941. URL http://portal.acm.org/citation.cfm?doid=1658939.1658941
20. Johnson, S.: Mobile broadband users seen hitting 1 billion in 2011 (2011). URL http://www.reuters.com/article/2011/01/11/us-ericsson-idUSTRE70A2JS20110111
21. Judd, G., Steenkiste, P.: Characterizing 802.11 wireless link behavior. Wireless Networks **16**(1), 167–182 (2008). DOI 10.1007/s11276-008-0122-5. URL http://www.springerlink.com/index/10.1007/s11276-008-0122-5
22. Keller, T., Olkin, I.: Combining Correlated Unbiased Estimators of the Mean of a Normal Distribution. Tech. rep., US Department of Agriculture and Stanford University, Stanford, CA, USA (2002). URL http://statistics.stanford.edu/ ckirby/techreports/GEN/2002/2002-05.pdf
23. Kernighan, B., Ritchie, D.: The C Programming Language, 2nd edn. Prentice Hall (1988)
24. Krifa, A., Barakat, C., Spyropoulos, T.: Message Drop and Scheduling in DTNs: Theory and Practice. IEEE Transactions on Mobile Computing (2011)
25. Krifa, A., Barakat, C., Spyropoulos, T.: MobiTrade: trading content in disruption tolerant networks. In: Proceedings of the 6th ACM workshop on Challenged networks, CHANTS '11, pp. 31–36. ACM (2011). DOI 10.1145/2030652.2030663. URL http://dx.doi.org/10.1145/2030652.2030663
26. Lacage, M.: Outils d'expérimentation pour la recherche en réseaux. Ph.D. thesis, Université de Nice-Sophia Antipolis (2010)
27. Lenders, V., Karlsson, G., May, M.: Wireless Ad Hoc Podcasting. 2007 4th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks **50**(2), 273–283 (2007). DOI 10.1109/SAHCN.2007.4292839. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4292839
28. Manshaei, M.H., Hubaux, J.P.: Hands-On Exercises: IEEE 802.11 Standard. Tech. rep., École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland (2010). URL http://mobnet.epfl.ch/hoe/HoE_1_IEEE80211.pdf
29. Nottingham, M., Sayre, R.: The Atom Syndication Format (2005). URL http://www.ietf.org/rfc/rfc4287.txt
30. Odlyzko, A.M.: Internet traffic growth: sources and implications. Proceedings of SPIE **5247**, 1–15 (2003). DOI 10.1117/12.512942. URL http://link.aip.org/link/?PSI/5247/1/1&Agg=doi
31. Project CCNx: CCNx Protocol (2012). URL http://www.ccnx.org/
32. Wallsten, S., Riso, J.: Residential and Business Broadband Price. Tech. rep., Technology Policy Institute (2010). URL http://works.bepress.com/scott_wallsten/59/
33. Warthman, F.: Delay-Tolerant Networks (DTNs): A Tutorial. Tech. rep., Warthman Associates (2003). URL http://www.ipnsig.org/reports/DTN_Tutorial11.pdf

Appendix A
Combining Unbiased Rate-Estimates

If we have multiple unbiased estimates of the request or delivery rate for the same content, we can compute a combined estimate as the weighted sum of each approximate value. We now focus our discussion on the request rate, although the following reasoning is also valid for combining delivery-rates. Let $\hat{q}_1^{(i)}, \ldots, \hat{q}_m^{(i)}$ denote m unbiased estimates of the request rate for content i ($q^{(i)}$). Now, define the combined estimate of i's request rate as follows.

Definition 6 *Let $\breve{q}^{(i)}$ be the combined estimate of m unbiased request-rate estimates for content i, $\hat{q}_1^{(i)}, \ldots, \hat{q}_m^{(i)}$. Then, $\breve{q}^{(i)}$ is defined as*

$$\breve{q}^{(i)} = \sum_{j=1}^{m} w_j \hat{q}_j^{(i)} \ , \tag{A.1}$$

where $w_j \geq 0$ denotes the weight of the j-th unbiased estimate and $\sum_{j=1}^{m} w_j = 1$.

We can show that the combined estimator has the following properties:

Lemma 5 *The combined estimator for the request-rate of content i, $\breve{q}^{(i)}$, is unbiased.*

Proof. Because the weights w_j and the unbiased estimates $\hat{q}_j^{(i)}$ are independent, for all $j \in [1, m]$, we have

$$\begin{aligned}
\mathrm{E}[\breve{q}^{(i)}] &= \mathrm{E}[\sum_{j=1}^{m} w_j \hat{q}_j^{(i)}] \\
&= \sum_{j=1}^{m} \mathrm{E}[w_j \hat{q}_j^{(i)}] \\
&= \sum_{j=1}^{m} \mathrm{E}[w_j]\,\mathrm{E}[\hat{q}_j^{(i)}] \\
&= q^{(i)}\,\mathrm{E}[\sum_{j=1}^{m} w_j]
\end{aligned}$$

$$\begin{aligned}\mathrm{E}[\breve{q}^{(i)}] &= q^{(i)}\,\mathrm{E}[\sum_{j=1}^{m} w_j] \\ &= q^{(i)} \ .\end{aligned}$$

Lemma 6 *Assuming that the m unbiased estimates of the request-rate of content i are independent, then the variance of the combined estimator for the request-rate for i is given by*

$$\mathrm{Var}[\breve{q}^{(i)}] = \sum_{j=1}^{m} \mathrm{E}[w_j^2]\sigma_j^2 \ , \tag{A.2}$$

where σ_j^2 is the variance of the j-th estimate of the request-rate for i.

Proof. We know that $\mathrm{E}[\breve{q}^{(i)}] = q^{(i)}$ and that the weights are independent of each request-rate estimate. By definition, the variance of $\breve{q}^{(i)}$ is given by

$$\begin{aligned}\mathrm{Var}[\breve{q}^{(i)}] &= \mathrm{E}[(\breve{q}^{(i)} - q^{(i)})^2] \\ &= \mathrm{E}[(\sum_{j=1}^{m} w_j \hat{q}_j^{(i)} - q^{(i)})^2] \\ &= \mathrm{E}[(\sum_{j=1}^{m} w_j (\hat{q}_j^{(i)} - q^{(i)}))^2] \\ &= \sum_{j=1}^{m} \mathrm{E}[w_j^2]\,\mathrm{E}[(\hat{q}_j^{(i)} - q^{(i)})^2] + \sum_{k\neq j} \mathrm{E}[w_j w_k]\,\mathrm{E}[(\hat{q}_j^{(i)} - q^{(i)})(\hat{q}_k^{(i)} - q^{(i)})] \ .\end{aligned}$$

As the m estimates of the request-rate are independent, $\mathrm{E}[(\hat{q}_j^{(i)} - q^{(i)})(\hat{q}_k^{(i)} - q^{(i)})] = \mathrm{E}[(\hat{q}_j^{(i)} - q^{(i)})]\,\mathrm{E}[(\hat{q}_k^{(i)} - q^{(i)})]$. Additionally, we know that the m estimates are unbiased, so $\mathrm{E}[(\hat{q}_j^{(i)} - q^{(i)})] = 0$. Using this result, we obtain

$$\begin{aligned}\mathrm{Var}[\breve{q}^{(i)}] &= \sum_{j=1}^{m} \mathrm{E}[w_j^2]\,\mathrm{E}[(\hat{q}_j^{(i)} - q^{(i)})^2] \\ &= \sum_{j=1}^{m} \mathrm{E}[w_j^2]\sigma_j^2 \ .\end{aligned}$$

By setting $w_j = 1/m$, for all $j \in [1,m]$, we obtain the **average combination estimator** [7], as defined next.

Definition 7 *The average combination estimator of the request-rate for content i is defined as the arithmetic mean of the m unbiased request-rate estimates $\hat{q}_1^{(i)}, \ldots, \hat{q}_m^{(i)}$:*

$$\breve{q}^{(i)} = \frac{1}{m}\sum_{j=1}^{m} \hat{q}_j^{(i)} \ . \tag{A.3}$$

This estimator is unbiased because the weights $w_j = 1/m$ and the unbiased estimates $\hat{q}_j^{(i)}$ are independent, for all $j \in [1,m]$. Hence, $\mathrm{E}[\check{q}^{(i)}] = m^{-1}\sum_{j=1}^{m}\mathrm{E}[\hat{q}_j^{(i)}] = q^{(i)}$. Additionally, the variance of the average combination estimator is given by $\mathrm{Var}[\check{q}^{(i)}] = m^{-2}\sum_{j=1}\sigma_j^2$, where σ_j^2 is the variance of the request-rate estimate $\hat{q}_j^{(i)}$. Although simple to compute, this estimator is susceptible to estimates with high variance as it weighs each combined estimate equally.

An alternative strategy to the average combination estimator consists in choosing weights proportional to the inverse variance of each estimate.

Definition 8 *Let the combined estimator for the request-rate of content i, $\check{q}^{(i)}$, be defined as the weighted sum of m unbiased request-rate estimates for i*

$$\check{q}^{(i)} = \sum_{j=1}^{m} w_j \hat{q}_j^{(i)} \ ,$$

such that the weights w_j are given by

$$w_j = \frac{\sigma_j^{-2}}{\sum_{k=1}^{m}\sigma_k^{-2}} \ . \tag{A.4}$$

For $m = 2$, the variance of the combined estimator defined previously is given by (A.2): $\mathrm{Var}[\check{q}^{(i)}] = (\sigma_1^2\sigma_2^2)/(\sigma_1^2+\sigma_2^2) \leq \min\{\sigma_1^2,\sigma_2^2\}$; hence, by combining two rate estimates we can reduce the variability of our original approximations. In practice, as the variance of a request-rate estimate is not known in advance, the weights w_j must be replaced by estimates that are *independent* of the request-rates being combined [7].