DYNAMIC ADAPTATION OF TRANSMISSION MODES FOR OPPORTUNISTIC CONTENT-CENTRIC NETWORKS

Masterarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Jürg Weber
2015

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik

Contents

List of Figures

List of Tables

Acknowledgment

First of all I thanks my supervisor Carlos Anastasiades for sacrifice his valuable time, for his detailed and patient explanations and his fabulous support. Prof. Dr. Torsten Braun for the opportunity to write a thesis in a practice oriented research group. Further my thanks goes to my office mates Tobias Schmid and Alexander Striffeler, for interesting coffee breaks, excitatory mini table tennis sessions and mind expanding ”FüBi’s” (after work beer’s).

Abstract

This thesis extend the content-centric network framework CCNx with Dynamic Unicast capabilities. Dynamic Unicast configures short term unicast routes to available content sources based on overheard broadcast Data messages. Two different forwarding strategies, namely Single Faces Forwarding (`SFF`) and Parallel Face Forwarding (`PFF`), have been implemented. These strategies supporting the Dynamic Unicast mechanism in mobile multi-hop environments.

Extensive evaluations in mobile environments with multiple requesters and content sources have been performed with NS3-DCE. We have shown that Dynamic Unicast performs in all scenarios better than broadcast. Significant throughput increases up to a factor of 37.3 have been measured. At the same time, because messages are directed only to required nodes (not flooded all nodes), forwarder nodes send up to 83.07% fewer Interest and up to 91.67% fewer Data messages.

Chapter 1

Introduction

1.1 Motivation

Content-centric networking (CCN) [1] is a new networking approach in which forwarding is based on content names instead of host identifiers. The main goal is to provide a communication infrastructure that is better suited to content distribution and mobility. Through caching and broadcast communication, CCN is more resilient to disruptions and failures. In earlier works [2], we evaluated that wireless unicast communication is more efficient than wireless broadcast because of higher throughput, lower message overhead, higher energy efficiency due to nodes quickly switching back to idle or sleep modes as well as automatic retransmission of lost packets. However, in a dynamic environment, static routes cannot be configured because neighboring devices are unknown. The goal of this master thesis is, therefore, to seamlessly switch between broadcast and unicast communication. Initial Interest requests are transmitted by broadcast to quickly find an available content source that answers the request by broadcast. A requester can then extract the identity, i.e., the IP address, from the received Data packet to configure a new unicast face. Future requests can then be directly forwarded to the content source by unicast. If the requester or the content source moves away, the Interest will timeout and broadcast requests are automatically used again as fallback.

The goal of this thesis is to implement this Dynamic Unicast scheme in CCNx 0.8.2 and evaluate in mobile (multi-hop) environments. The evaluation investigates the robustness of Dynamic Unicast compared to broadcast in such mobile environments, concerning messages and retrieval time.

1.2 Task and Problem Formulation

In CCNx, packets are routed based on their name-prefix and not based on host identifiers as in IP-networks. The main focus of the CCNx development points to wired Internet where name-prefixes do not change often. At the moment the CCNx forwarding tables (FIB) are populated manually at start-up. During operation, no forwarding table updates are provided. This is not feasible for mobile networks where connections and available content may change often. Due to

changing communication partners, FIB entries cannot be set up statically and messages need to be transmitted by broadcast. Information about available content can be obtained by overhearing broadcast communication. However, previous work [2] has shown that broadcast communication is inefficient in terms of message overhead and transmission speed. Thus, optimizations are required to increase communication efficiency. An approach is extract node IDs (e.g., IP addresses) from incoming broadcast packets to configure short-term unicast links to neighbors that can provide content. The tasks of this master thesis are listed below:

- Implementation of an IP source extractor in C. This extractor extracts the source IP address from broadcast Data packets and create a new unicast face to the sender.
- Implementation of a CCNx forwarding strategy for mobile environments.
- Implementation of an efficient update mechanism in case of path breaks.
- Implementation of a data structure which keeps track of concurrent unicast requests at a source. Replace multiple concurrent unicast requests with a broadcast transmission for higher traffic efficiency. In particular, find a threshold R and evaluate options to switch from multiple unicast transmissions to broadcast.
- Evaluation of the implementation in a mobile scenario using the network simulator `NS3-DCE` [3].

1.3 Outline

The remainder of this report is structured as follows: In Chapter 2 we describe the content-centric networking approach applied in the CCNx framework. Design and Implementation details of the Dynamic Unicast functionality and the `CRT table` are described in Chapter 3. Chapter 4 describes the evaluation environment and presents our evaluation results. Finally, in Chapter 5, we conclude our work and give an outlook to future work.

Chapter 2

Related Work

2.1 Content-Centric Networks

Content-centric networking (CCN)[1] is a new networking approach. In content-centric networks, routing is based on content names instead of host identifiers. There are two CCN packet types, namely Interest and Data.

2.1.1 Interest and Data messages

Figure 2.1 illustrates an Interest and Data message.

The Interest Message is used to request Data by name. Content is organized in segments and Interest messages are used to retrieve specific segments directly. Interest messages have a name prefix (for the content) and can have additional header fields to restrict data retrieval such as Answer Origin Kind (AOK) and scope. AOK allows to alter the usual response to Interest, e.g. avoid answering Interests from content store (cache). The scope limits where the Interest may propagate, for example scope 2 means, propagate no further than the next host.

The Data message is used to supply data. Besides the segment (data payload), a Data message also contains a cryptographic signature to ensure authenticity of the data (publisher's signature). Every Data message is required to contain a valid signature. Each content file consists of many Data messages, and each of these objects is identified by a segment number.

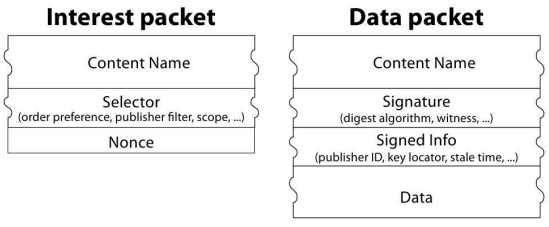


Figure 2.1: Interest and Data packet. A consumer needs to express an Interest in a content name to receive the corresponding data packet [1].

2.1.2 CCN Forwarding

The forwarding engine is included in the CCNx daemon (`ccnd`, see 2.2.1). Figure 2.2 illustrates a schematic overview. The CCN forwarding is based on three basic components:

- Pending Interest Table (PIT)
- Forwarding Interest Base (FIB)
- Content Store (CS)

If an Interest is received, the CS is checked, whether a matching object is available. If not, the PIT is considered whether already a similar Interest is pending. If there is already an entry in the PIT, the Interest can be discarded because an answer is already pending. If there is no entry in the PIT, the FIB holds information where to forward the Interest. If no information is available in the FIB, the Interest is discarded.

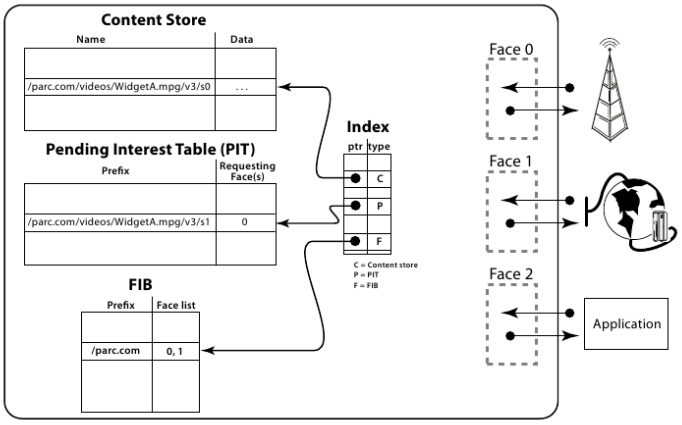


Figure 2.2: CCN forwarding engine model [1]

2.1.3 Naming

Names are composed of multiple components structured in a hierarchical way. An individual name is composed of a number of components. Each component is separated with a slash "/". This delimiters are only for human readability but not part of the name (names are binary encoded). Figure 2.3 shows a name in the human readable as well as the binary encoded from used in the framework. These strings are called name prefixes.

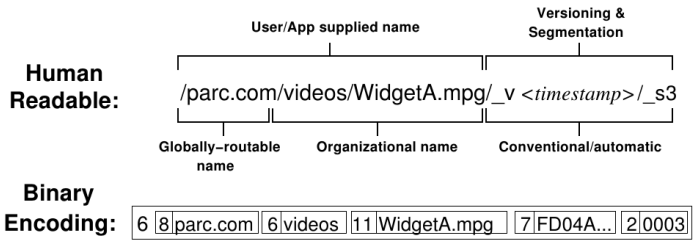


Figure 2.3: Example Data name [1]

2.1.4 CCN Faces

In CCN, links to other hosts or applications are called faces. In case of applications, the face corresponds to local Unix sockets, in case of hosts, it corresponds to UDP or TCP sockets. Therefore, multiple faces can be supported over the same interface.

The forwarding of Interests in CCNx is based on the FIB. Every entry in the FIB contains a prefix which directs Interests based on longest-prefix match and a face where to forward it. Every entry is only valid for a specified (prefix) lifetime. If the prefix lifetime expires or the FIB entry does not contain a valid face (anymore), the FIB entry is removed.

2.2 Project CCNx

CCNx [4] is an open source project providing an implementation of the content-centric networking concept (CCN). The work performed in this thesis is based on CCNx. The framework is written in C.

2.2.1 CCNx Daemon

The CCNx daemon, called `ccnd` [5], is the central component of the CCNx implementation. It is running on every host and it performs all processing and forwarding of received messages. The typical configuration is to run one `ccnd` on each host. Applications running on the host will communicate via local faces with the `ccnd`, which may redirect Interests to attached networks based on the FIB configuration.

2.2.2 Routing Daemon ccndc

The routing daemon `ccndc` [6] is a utility to configure the FIB, i.e., adding or deleting static FIB entries. FIB entries are added by specifying the prefix and face information. A face is defined by host address, port number, and transport protocol (UDP or TCP). In this work we only consider the UDP protocol.

2.2.3 CCNx key management

Integrity and authenticity in CCNx is ensured by signing every Data message with the private key of the publisher. For verifying a signature, the public key of the signer must be known. The public keys are stored under the prefix `ccnx:/ccnx.com/Users`. It must be ensured that this prefix is stored as a forwarding entry in the FIB to retrieve keys if they are not locally available.

2.2.4 CCNx Flags

The CCNx daemon and CCNx applications uses many different flags for face and message processing. In principle a CCNx flag describes a state or a specific setting of an arbitrary CCNx functionality such as a face or a message. Below a short description of the flags which are used within this thesis.

- CCN_FACE_LOCAL
  Face is mapped to localhost address.
- CCN_FACE_GG
  Face is considered as friendly, which means it is mapped to a local application.
- CCN_FACE_MCAST
  Defines whether a face is mapped to a multicast address.
- CCN_FORW_ACTIVE
  Defines whether a face is active (allowed to forward data).

More information about specific flags can be found on the CCNx Face Management and Registration Protocol man page [7].

2.2.5 Pipelining

The pipeline size defines the number of segments that can be concurrently requested.

2.2.6 CCNx Strategies

Unlike in IP, CCNx FIB entries may point to multiple next-hop destinations simultaneously. The self-identifying nature of Interest Messages and Data messages means that loops can be avoided without establishing a spanning tree allowing only one destination for any particular prefix at any particular node. The possibility of multiple outbound faces means that different strategies may be employed to select among them when forwarding messages [8].

Each node must implement at least one strategy rule to forward Interest and Data messages. To increase routing efficiency for different situations different strategies may used.

2.2.7 CCNx Overhearing

Overhearing is the key principle for applying Dynamic Unicast. In previous work [2], we have defined overhearing as follows:

> Overhearing is the activity of a network participant listening to the communication between two other participants. Using a broadcast face is the only way to perform overhearing with CCNx.

The content prefix can be extracted from the received Data message and configured to the existing broadcast face in the FIB. The entry can contain a short expiration time, which is updated whenever content with the same prefix arrives. Second, prefixes expires if no data arrives on the face anymore. Message processing in [2] has been performed in the following way. First a content prefix will be added (or updated if it is already available) with an expiration time if the content has been overheard on a specific face. Second, prefixes expires if no data arrives on the face anymore. However, evaluations have shown [2] that broadcast communication is very slow and produces a massive message processing overhead (because all nodes, independent whether they requested content or not, need to process the messages). Thus, instead of broadcast communication, in this work we use Dynamic Unicast, which will be introduced in Section 3.6, to create short term unicast faces to neighbor nodes. Such a face is created based on overheard broadcast Data messages and has only a limited validity. In case of inactivity the face will be removed from the FIB. Following content requests will be forwarded over broadcast till Dynamic Unicast creates a new short term unicast face. By that, nodes in mobile environment can benefit from faster transmission times and a lower message processing overhead (lower energy consumption).

2.3 Reimplementation From Former Work

In previous work [2] name prefixes were extracted from overheard broadcast Data transmissions to configure FIB entries. Then, Interest messages could be forwarded via broadcast face over multiple hops. To enable overhearing in the first place (without configured FIB entry), Interests from local applications were always transmitted on a broadcast face (if no other face was configured). We refer to this as `pass-through` of local Interests. To limit the number of forwarding entries in the FIB, a `prefix cut` has been implemented, which identifies common prefixes in multiple names. By this, a more general (common) prefix can be used representing multiple overheard content names. The evaluation has been performed on three Alix3D2 mesh nodes [9], which have been placed near each other on a table. To emulate different transmission range, different channels have been used. Evaluations have shown that broadcast communication leads to extensive flooding in the network. In particular, at the end, every node in the network ends up having broadcast prefixes configured in the FIB. Furthermore, broadcast communication has shown very low data rates.

In contrast to previous work [2], not only prefixes but also node IDs, i.e., IP addresses, have been extracted from overheard Data transmissions. This information is used to create

Dynamic Unicast faces towards content sources enabling more structured routing (no flooding). Furthermore, `pass-through` forwarding of local Interests has been extended for multi-hop communication. Two forwarding strategies have been defined for single and multi-path routing. In addition, a content request table (CRT) has been introduced to identify multiple incoming unicast requests such that Data can be returned via one broadcast instead of multiple unicast transmissions. The evaluations has been performed with NS3-DCE [3], which enabled more complex and dynamic scenarios with more than 100 (mobile) nodes in contrast to the table experiments in previous work [2]. An evaluation script, written in python and bash has been implemented to automate processing of the vast number of evaluation results.

However, this thesis still uses some basic principles from previous work [2]. The list below describes the reused parts and its differences to [2] (in *italic*):

- **Messages**: Only Data messages are overheard and no Interest messages because Interests should be forwarded based on the availability of Data. This principle is reused in this thesis. *In contrast to [2], this thesis uses the overheard content information (node ID and content name) to register new unicast connections, whereas previous work [2] has used only the overheard content name to register new broadcast prefixes in the FIB.*

- **Entry Lifetime**: An overheard prefix entry should only be valid for a short time. In mobile environments communication partners can change fast and frequent. *In this thesis the prefix lifetime (validity period) is set to 5 seconds, which is 1 second more than the default Interest lifetime of 4 seconds. Thus, cleanup of outdated FIB entries can be done frequently and quick. In addition to the prefix entry lifetime, face activity is periodically checked and inactive (dynamically created) unicast faces are automatically removed. This strategy exploits existing mechanisms in CCNx for dynamically created unicast faces, which are created to return Data if requests have been received via unicast.*

- **Prefix registration**: In previous work [2], a prefix cut has been performed to limit the number of entries in the FIB. This prefix cut allows a certain (by users choice) maximum prefix length. For two prefixes of different Data messages, e.g. `ccnx.org/my/data/thisplace/data1` and `ccnx.org/my/data/otherplace/data2`, the common prefix `ccnx.org/my/data` can be extracted and configured in the FIB. *In this work, the complete name (base name without segment number) is configured (no prefix cut) because configured FIB entries are only valid for a short time, i.e., only for a specific stream. Furthermore, overheard prefixes are registered on a unicast face.*

- **Broadcast pass-through**: The existing implementation enabled the forwarding of local Interests to a broadcast face. *The implementation has been extended such that also received Interests from neighbor nodes can be forwarded to a broadcast face if no matching unicast face is available (to enable multi-hop communication).*

Chapter 3

Design and Implementation of Dynamic Unicast

This section describes the implementation of Dynamic Unicast. All described functionality is integrated directly in the CCNx message forwarding daemon (`ccnd`).

3.1 Overview

The goal of Dynamic Unicast is to perform broadcast communication only if necessary. Therefore, if a Data message has been received via broadcast in response to a transmitted Interest, the node identifier from the sender (previous hop) is extracted to create a direct (unicast) face to the node. In this thesis, we use the IP address as node identifier. This process is further detailed in subsection 3.5. The next Interest transmissions can then be forwarded via the direct (unicast) face to the next hop towards a content source and do not need to be broadcasted again. In Section 3.2 we describe how long such forwarding entries need to remain valid. This is particularly important to avoid many invalid unicast faces in dynamic mobile environments because they would make message forwarding very inefficient since there would be many faces to not reachable sources. Section 3.3 describes an option to enable and disable Dynamic Unicast. Section introduces the concept of broadcast `pass-through`. Section 3.6 gives an overview over the entire Dynamic Unicast process and with all of its components. Finally, Section 3.7 introduces an optimization for highly requested content in static networks, the so called `Content Requester Tracker` (`CRT`), which allows the content source to switch from multiple unicast connections to broadcast.

3.2 Lifetime of Dynamic FIB entries

CCNx provides a mechanism to remove dynamically created faces (the function `reap()`) if no content has been received over that face for 4 times the Interest lifetime, i.e., 16 seconds. This is a long value for very mobile environments, where neighbors may change quickly. Since forwarding via outdated forwarding entries can result in very inefficient forwarding, we adapted the automatic face removal process and delete faces already if no content has been received

within 4 seconds, i.e., 4 times faster than the default value. For convenience, we introduce an environment variable `CCND_REAP_FACTOR` to modify the deletion time. If the value is set to 0 it is the default, each increase by 1 halves the value.

Nevertheless, it is still possible that a node receives Data messages for another content (other name prefix) over the same face. Therefore, it is not possible to delete the face. To solve this problem each overheard name prefix has only a short FIB lifetime. Currently the value is set to 5s, which is slightly higher than Interest lifetime of 4s. As long as prefix related content arrives on the face, prefix lifetime will be updated, otherwise the name prefix expires and no forwarding is possible anymore.

3.3 Enable and Disable Dynamic Unicast

Since broadcast adaption may not be required in static settings (because it can be configured manually), we introduced the environment variable `CCND_UNIA` to disable Dynamic Unicast and all the processing described in this thesis. To enable Dynamic Unicast, `CCND_UNIA` need to be set to 1, otherwise it is disabled and `ccnd` operates in the standard way.

3.4 Broadcast Pass-Through

Broadcast `pass-through` (as implemented in previous work [2]) enables the forwarding of local Interests over a broadcast face if no matching forwarding entry can be found in the FIB. To enable this, the exact content name (without segment number) is configured to the FIB for a short time (in our implementation: 5s). If content is returned, lifetime of the FIB entry can be extended.

Broadcast communication requires the configuration of two broadcast faces such that Interests received over one face can be forwarded over the other face. In this work, we have extended broadcast pass-through for multi-hop communication such that Interests from other requesters received over one broadcast face can be forwarded via the other broadcast face in the absence of a matching FIB entry.

3.5 Overhearing with Dynamic Unicast

Figure 3.1 illustrates overhearing process of Dynamic Unicast 3.6, which has been implemented by adding functionality to three functions in the `ccnd`, i.e. `process_input()`, `process_input_message()` and `process_incoming_content()`. Each step is denoted with a number. The following description references to these numbers.

First, `process_input()` is called with an incoming message and the related socket file descriptor (1 and 2) from the lower layer. If Dynamic Unicast is enabled, i.e., `CCND_UNIA`

set to 1 and the message has been received at a broadcast face (3), the socket information is retrieved additionally (4). After that (or without Dynamic Unicast), the message is parsed from its binary form to the ccnb encoded (5) and further processed in `process_input_message()`.

In `process_input_message()`, the message is further processed based on its message type i.e., Interest, Data message or protocol message (6). If it is a Data message, `process_incoming_content()` is called. Please note that we only consider Data message for overhearing (Dynamic Unicast), because they are sent by the content source and, thus, indicate the path to the content source. All other messages are processed according to the `ccnd` standard procedures (12).

The function `process_incoming_content()` describes how to handle incoming Data messages. To apply Dynamic Unicast, the content has to have at least one matching Interest (7). We do not perform overhearing of content without Interests i.e., not requested content, to limit the entries in the FIB. Additionally, FIB entries may expire quickly in mobile environments, thus, it may be not efficient to process and maintain routes to other content sources, since outdated routes may degrade the forwarding performance. If there is an Interest match (7), Dynamic Unicast registers a new unicast face based on the stored socked information if it does not already exist. Faces that are registered dynamically due to Dynamic Unicast will be flagged with `CCN_FACE_UNIA`.

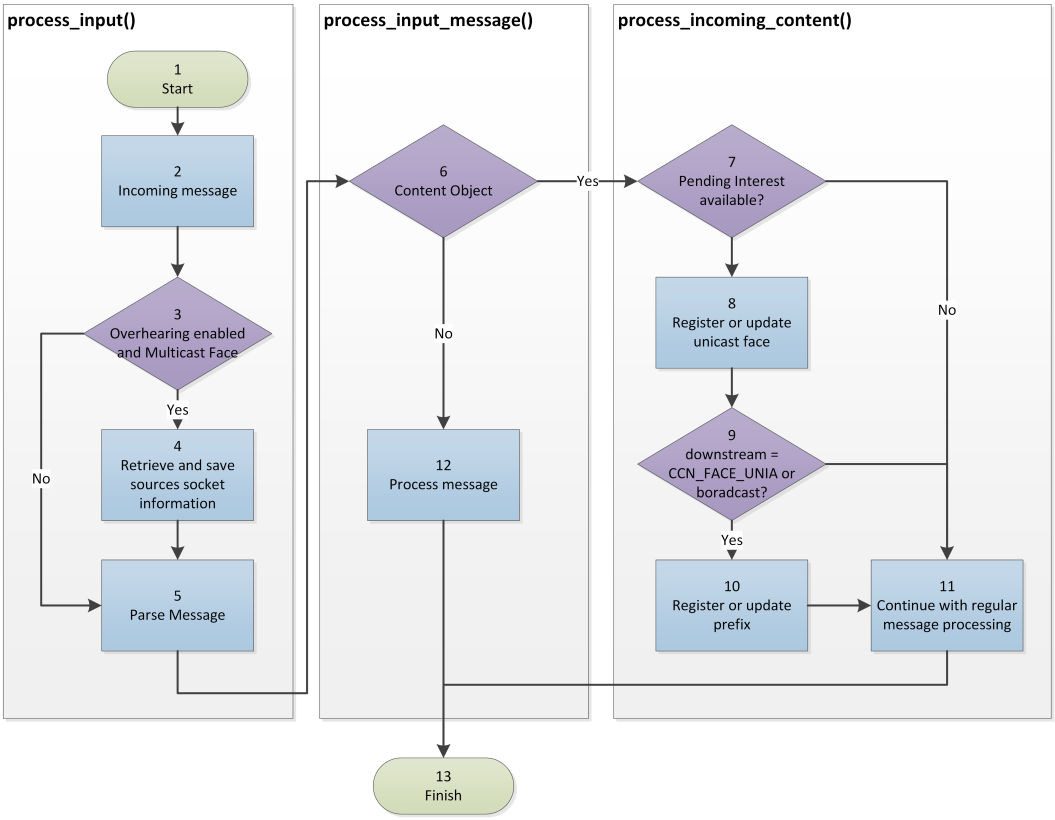


Figure 3.1: Overhearing with Dynamic Unicast

If the face existed already before, face information is updated indicating that content is received over that face. This process is already included in CCNx and supports to delete dynamic faces, which do not receive content over a certain time (8). See subsection 3.2 for more information. In case of a new face registration, the content prefix needs to be registered in the FIB table (if it does not exist yet) and associated with that face (9 and 10). We consider the complete content name (without segment numbers) as content prefix in the FIB. Please note that dynamic FIB entries are limited due to their short validity and the fact that content is only overheard if a corresponding Interest has been transmitted. Dynamically created FIB entries (they have the `CCN_FACE_UNIA` flag) have a short validity time in the FIB (in our implementation: 5s) but they are updated whenever new content has been received (no duplicates). This concludes Dynamic Unicast and the message can be further processed in the regular way (standard implementation, 11).

3.6 Dynamic Unicast Process

Figure 3.2 illustrates the Dynamic Unicast process. The implementation of Dynamic Unicast in the `ccnd` (right part in Figure 3.2) interacts with applications (left part in Figure 3.2). The application part is executed by a retrieval application (we will give some examples below),

while the `ccnd` part describes what happens at the `ccnd` (Dynamic Unicast). Each process is denoted by a number, which we use as reference in the following description. The `strategy callout` describes the forwarding strategies, which are explained in more detail in section 3.6.2.

As explained above, the `ccnd` daemon interacts with applications, e.g., for content retrieval. The following applications are examples of such requester applications.

- **ccncat** [10] - This is a standard content retrieval application in CCNx 0.8.2. For a given content name, the application performs a version resolving and then starts content retrieval with the corresponding version. This application is not efficient in mobile and opportunistic networks because of two reasons. First, version resolving is only performed once, i.e., if it fails, the file transfer is aborted. Second, if file transfers are disrupted, file transfers can not be resumed but need to start again from the beginning. Thus, in case of short contact durations, file transfers can never be completed.

- **ccnacat_resume** - This requester application supports resume capabilities and longer version resolving to enable content downloads in opportunistic environments with short contact durations. This requester application has been developed within the scope of another master thesis [11]. This application is advantageous in mobile environments, where transmission may be disrupted frequently.

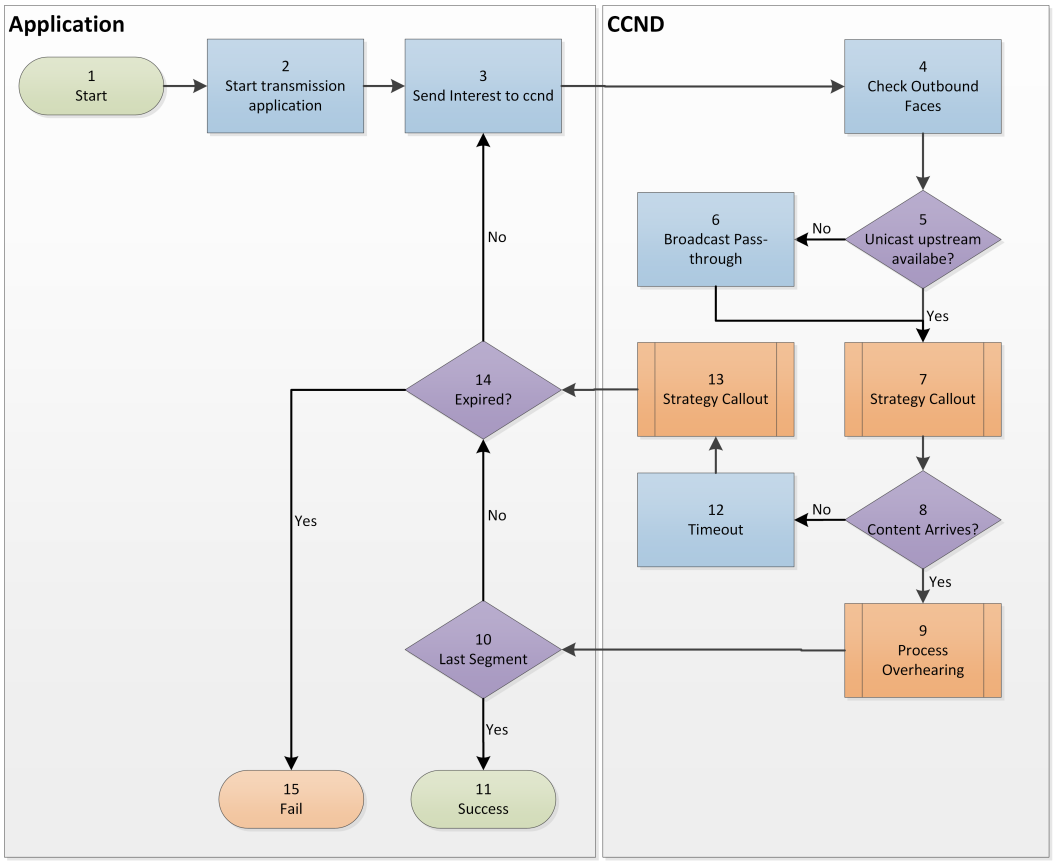


Figure 3.2: Dynamic Unicast process. Left part illustrates a requester application such as ccnacat_resume, the right part illustrates the local `ccnd`.

Message processing in Figure 3.2 starts by an application initiating a file transmission of specific data message (2). In a first step, version resolving is done to find a version number of the requested content name (not shown in the figure for simplicity). After the version has been found, Interests will be send to the local `ccnd` (3). `Ccnd` is the ccn daemon and responsible for message processing, forwarding and also Dynamic Unicast. In a first step the `ccnd` checks whether outbound faces for the requested content name are available (4). If there are no matching unicast faces (5), the Interests are forwarded over broadcast. The `broadcast pass-through` (see Section 3.4) ensures that communication is possible even when no unicast face is available. Therefore, Dynamic Unicast creates a new prefix entry to a given broadcast face (6). However, if there is at least one unicast face with a matching prefix available, the Interest can be sent over it. Dynamic Unicast prioritizes unicast connections, thus, if there is a matching unicast face, it will be preferred over a broadcast face. Then, the `strategy_callout` (7) is called with `CCNST_FIRST`, which implies that a new Interest is transmitted. Please refer to Subsection 3.6.2 for more information on `strategy_callout`. The `strategy_callout` starts a background job in the daemon, which will return with `CCNST_TIMEOUT` if an Interest timeout occurs. However, if the requested content arrives

in time at the daemon (8), the overhearing process is initiated (9) to extract the content prefix and register it into the FIB. Please refer to subsection 3.5 for more details on the overhearing process. Then, the content is forwarded to the application. In this thesis, we use the `ccnacat_resume` application, which requires content retrieval to be finished within a specified time (expire time), e.g., 3 hours. Thus, if the maximum content retrieval time (expire time) has been reached, the application considers the content retrieval as failed (15). If it is the final segment, and all previous segments have been received, the application ends with a success (11).

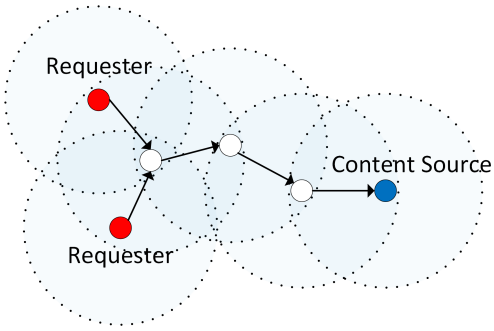
In case of a timeout (12), i.e., content is not returned in time, the daemon performs a strategy callout (13) with `CNST_TIMEOUT`, which implies an Interest timeout. Interest timeouts are handled by the application. First, the application checks whether maximum content retrieval time (expire time) has not been expired (14). If so, the Interest is re-expressed and send to the `ccnd` (3). if the content retrieval expire time is reached (13), the request will be considered as failed (15).

3.6.1 Multi-hop with Dynamic Unicast

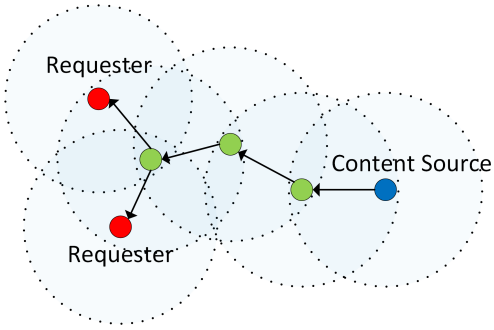
In this Subsection we describe how Dynamic Unicast works for multi-hop communication.

Figure 3.3 illustrates a network with two requesters, a content source and multiple forwarder nodes. Requesters broadcast Interest messages, which are received and forwarded by other nodes (Figure 3.3a). Forwarding without configured FIB entries is possible due to the broadcast pass-through (see Section 3.4). Each forwarder node has two broadcast faces for multi-hop communication. Multi-hop broadcast `pass-through` enables an Interest received over one broadcast face to be forwarded over the other broadcast face (if no prefix is configured in the FIB). Furthermore, as shown in 3.3a, Interests from multiple requesters can be aggregated in the PIT such that only one Interest needs to be forwarded further.

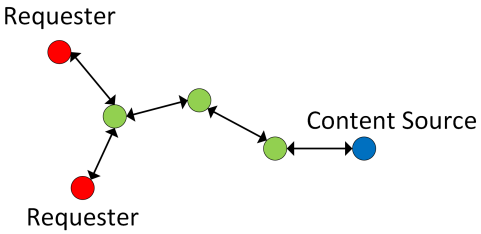
If an Interest reaches a content source, Data returns on the reverse path via broadcast as shown Figure 3.3b. By that all node that overhear the Data transmission can configure an unicast face to the previous hop in the FIB. After the Data packet has reached the requesters, a hop-by-hop unicast path to the content source is configured in the FIBs of all intermediate nodes. Thus, the content transfer can be continued via unicast (Figure 3.3c). The unicast path is valid as long as the nodes are in each others range. If the path breaks somewhere, e.g. due to node movement, some Interests will time out, which causes the Dynamic Unicast faces to be removed from the FIB. Interests are then retransmitted via broadcast face to find a new content source (implicit content discovery).



(a) Requesters flood Interest messages



(b) Content source replies Data via Broadcast. Nodes can register unicast face



(c) Data transmissions via configured unicast faces. Interest aggregation and caching is still possible.

Figure 3.3: Multi-hop with Dynamic Unicast

3.6.2 Forwarding Strategies for Dynamic Unicast

In CCNx 0.8.2, different forwarding strategies can be integrated. The forwarding strategy is selected in a `strategy_callout` as explained in subsection 3.5. The strategy can be triggered by `CCNST_FIRST` (first Interest) or `CCNST_TIMEOUT` (Interest timeout) events. To support Dynamic Unicast in mobile environments, two forwarding strategies, i.e., Dynamic Unicast with single face forwarding (`SFF`) and Dynamic Unicast with parallel faces forwarding (`PFF`), have been implemented.

Both strategies are explained by flow charts in the next Subsections. Each step has a number. The following explanations reference to those steps with numbers in brackets.

Single Face Forwarding (SFF)

Figure 3.4 shows the flow chart of `SFF`. The goal of `SFF` is to forward an Interest to only one face at a time. The forwarding process is started via a strategy callout. First, strategy related information is gathered, e.g., from which face the Interest came and further forwarding information for the name prefix such as state and a predicted response time (if available) (2). Based on the event that has triggered, i.e., in this thesis we only consider `CCNST_FIRST` and `CCNST_TIMEOUT`, the Interest is handled differently (3). In case of a `CCNST_FIRST` event,

an incoming Interest needs to be forwarded and the strategy decides, which outbound face to use. There is always at least one broadcast face to forward the Interest. However, if there are multiple unicast faces, the best unicast face will be selected. The best face for a prefix is defined as the face over a Data message has been successfully received before (4). If there is no best face, the most recently added unicast face is considered as best face. If there are no unicast faces, the broadcast face (fallback face) is considered as best face.

Then, the Interest is forwarded via the best face. As long as the Interest does not expire (content comes back), the chosen face will be considered as best for future strategy call outs. After the Interest has been forwarded, it is added to the pending Interest table and a timeout timer is started (5). If the pending forwarded Interest entry expires (after 4s), a `CCNST_TIMEOUT` occurs (7). The related name prefix will be removed from the face and the face will be deleted (if no other prefixes are registered). Since it is possible that there are other unused faces in the FIB, a unscheduled `reap()` call checks for outdated information and updates to the FIB table (8).

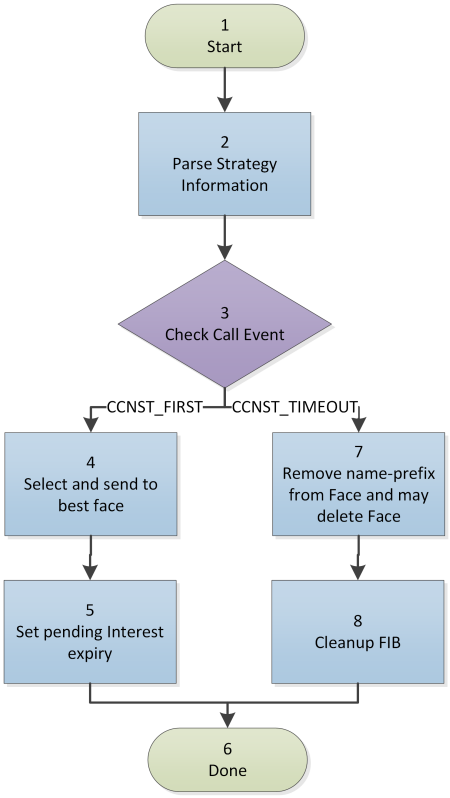


Figure 3.4: Single face forwarding strategy

Parallel Faces Forwarding (PFF)

Figure 3.5 illustrates the work flow of `PFF`. This strategy is a bit closer to broadcast, since it sends the Interest not only to one face but all available unicast faces in parallel (4). Thus, there is no need to define a best face. All other steps are similar to `SFF` 3.6.2.

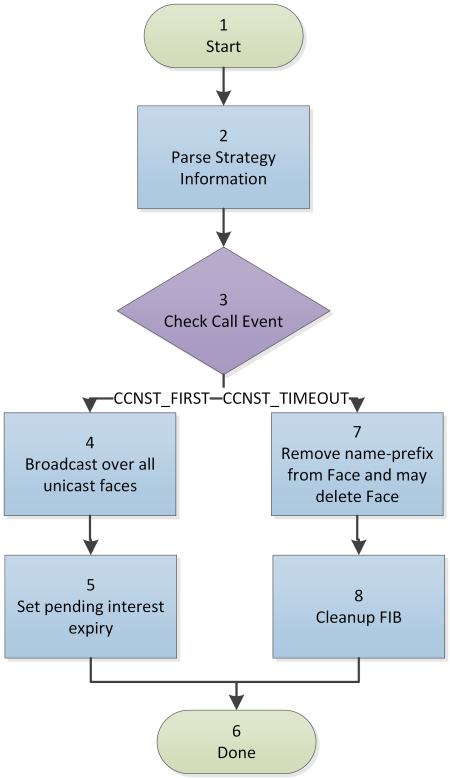


Figure 3.5: Parallel faces forwarding strategy

The reasoning behind the design of `PFF` is that in high mobile environments it may be advantageous to have different routes for redundancy to retrieve content quicker and more reliable if one route breaks. Thus, there is redundancy in message forwarding, which may be traded for higher robustness of content retrieval in mobile environments.

3.7 Content Request Tracker (CRT)

Dynamic Unicast is beneficial in static and mobile environments if there are a few requesters in the content. However, if there are many concurrent requesters in the same content, e.g., video streaming, a single broadcast transmission may be more efficient than many independent unicast transmissions. To address this case, we extended the implementation with a `Content Request Tracker` (`CRT`), which keeps track of the number of unicast content streams and performs a broadcast transmission instead of multiple unicast responses if a certain number of requests (threshold `R`) is reached.

3.7.1 Modification at the Content Source

Figure 3.6 shows the flow chart of a source node, which has `CRT` enabled (`CRT_ACTIVE` is set to 1). Since `CRT` considers the number of distinct unicast Interests to switch to broadcast, we only consider unicast Interests. Connection information is stored in a `CRT token`. The `CRT token` contains all necessary information about ongoing content downloads such as the content prefix and the number of unicast faces (to requesters) related to the prefix. Different `CRT tokens` (for different name prefixes) are stored in a hash table (keyed by the name prefix). `CRT tokens` have a validity time and they can be deleted due to two reasons. First, if the final segment is requested, the transmission is finished and the `CRT token` is removed from the hash table. Second, every `CRT token` has an additional expiration time (in our implementation set to 1min). After a period of inactivity (no requests for the expiry time), the `CRT token` expires and is removed from the hash table. However, `CRT tokens` can also be updated in case of new incoming Interests from different requesters (added to the connection list).

If the `ccnd` receives an incoming Interest on an unicast face, it has a lookup to the content store and, if not available, into the repository (2). If the content is available in the repository (3), the related `CRT token` will be updated or created if not available. Furthermore, the expire time of the `CRT token` will be updated since there are still connections to this content (4). The face over which the content source responds to Interests depends on the number of open unicast requests (streams) from different requesters. If the number of streams is higher than `CCND_MAX_CRT` (5) the content will be sent over broadcast (6a) otherwise each request will be satisfied via unicast (6b). After forwarding the Data message, a hash table lookup is done to get the related CRT-token. If the currently sent Data message has been the final segment, the `CRT token` will be removed form the hash table (9).

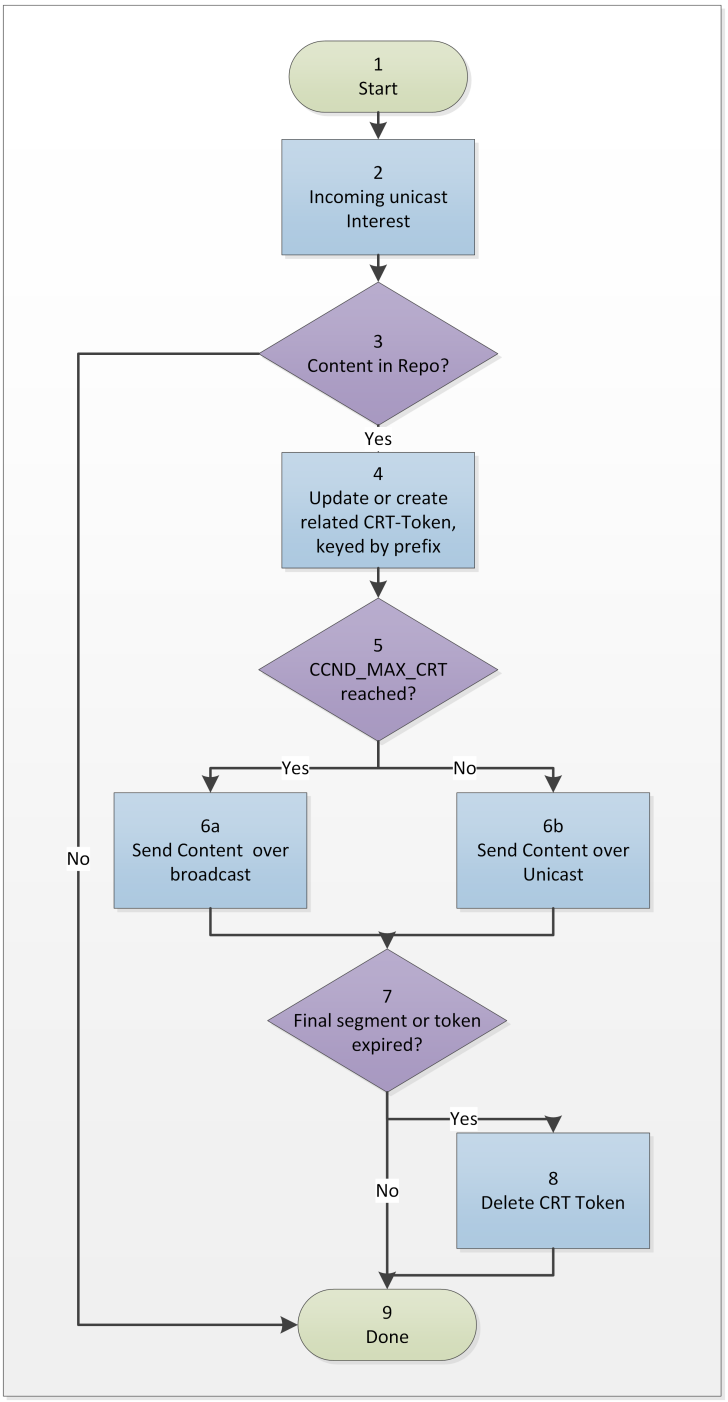


Figure 3.6: Flow chart CRT - source side

3.7.2 Modification at the Requester

If a requester began to send Interests via dynamically created unicast face (Dynamic Unicast) it will continue to send the Interests over this face, as long as content comes back from the same source (unicast entries are updated for new content). This may result in an inefficient behavior because there are many unicast requests for an increasing number of requesters. As a possible optimization, a requester should switch from unicast to broadcast when a certain number of subsequent unicast requests (in our implementation: 2) are answered via broadcast, which indicates that the content source has switched to broadcast (many requesters).

Figure 3.7 illustrates the required optimizations at the requesters for incoming Data messages (left flow chart) and incoming Interests (right flow chart). First the incoming Data message will be considered. If the face on which incoming data Message has been received is broadcast (2), the related broadcast content counter will be updated. This counter is keyed by the Data messages name prefix (3). Afterwards, the Data message will be processed according the `ccnd` standard procedure (4). If an Interest from a local application has been received (right side of Figure 3.7, 2), the related broadcast counter is considered before it is forwarded. If the counter is higher than the limit (e.g. `CCND_MAX_CRT_REQ`: 2) (3) the Interest will be sent over broadcast (4a) otherwise over unicast (4b). Afterwards, the Interest will be processed according the `ccnd` standard procedure (5).

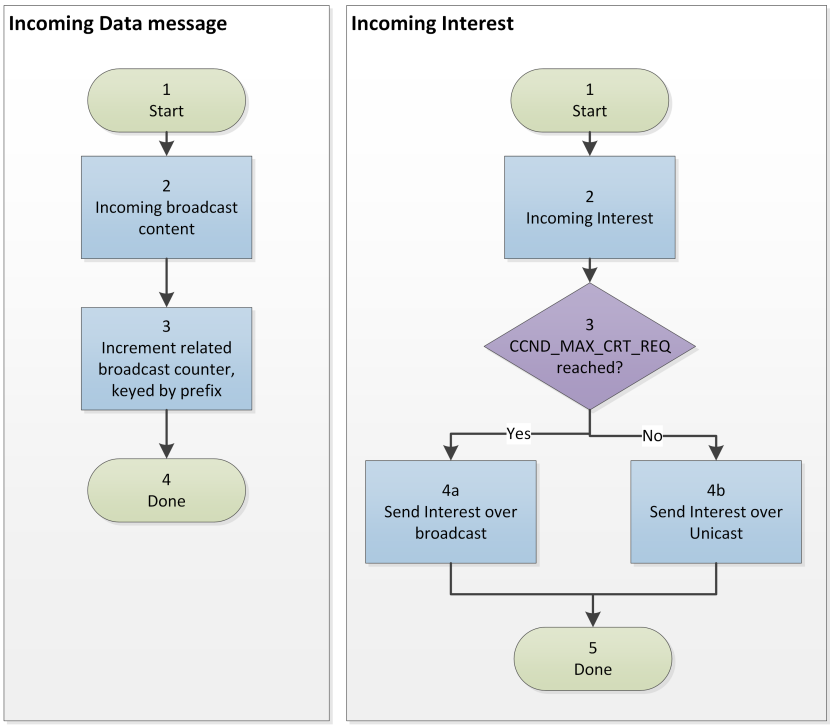


Figure 3.7: Flow chart CRT - requester side

Chapter 4

Evaluation

The main goal of this chapter is to evaluate and compare Dynamic Unicast against broadcast in static and mobile networks. First, we evaluate it in simple scenarios. Afterwards, we evaluate it in more complex mobile scenarios. We consider the following scenarios in our evaluations:

- **Scenario 1 - Multi-hop in a line topology 4.3**
  This scenario compares unicast and broadcast with Dynamic Unicast (`SFF`) in a line topology. To reach the content source, Interests from a requester need to be forwarded by multiple nodes. We will evaluate the overhead of Dynamic Unicast compared to preconfigured multi-hop unicast routes.

- **Scenario 2 - Multi-hop communication with static requester and source 4.4**
  In this scenario there is a static requester and a static source, which are multiple hops apart. Additionally, there are multiple mobile nodes between the static source and requester moving according to the Random Waypoint mobility model [12]. This evaluation shows the impact of mobile forwarding nodes, i.e. where routes do not hold for long. Please note that we perform forwarding only based on paths discovered via broadcast, i.e., we do not use geographic forwarding. Requester and content source are static to ensure multi-hop communication (if they would move as well, no multi-hop communication may be performed if they are in direct transmission range).

- **Scenario 3 - Single-hop and Multi-hop in a Grid Topology 4.5**
  In this scenario multiple content sources are statically placed in a grid. Multiple mobile nodes are moving according to the Random Waypoint mobility model and request content. We will evaluate scenarios with multiple concurrent requesters. Two sub scenarios will be evaluated, single-hop, and multi-hop.

- **Scenario 4 - Circle Mobility 4.6**
  This scenario is the same as in the Master thesis of Tobias Schmid [11]. It simulates a hiking round trip to a content source, which has no direct contact to the requester, e.g., a sensor on the top of a hill. The requester is in the valley and requests content with the help of hikers that forward the requests over multiple hops to the content source.

- **Scenario 5 - Static Circle 4.7**
  In this scenario, single hop unicast and broadcast will be compared for an increasing number of requesters on the circle (content source in the center). The goal is to define a threshold `R` where CCNx broadcast transmission is more efficient than CCNx unicast transmission. After defining this `R`, the concept of `CRT` can be evaluated.

4.1 Simulation Environment and Hardware

Dynamic Unicast has been implemented in CCNx 0.8.2 and evaluated with the NS3-DCE [3] version 1.4 network simulator.

> Direct Code Execution (DCE) is a framework for NS3 [13] that provides facilities to execute, within NS3, existing implementations of user space and kernel space network protocols or applications without source code changes.

Using NS3-DCE, it is possible to evaluate more complex scenarios than in testbeds with wireless mesh nodes. Hardware costs for testbeds with more than 100 nodes would be high and evaluation analysis would be more complex such that (repeatable) evaluations in large mobile scenarios become nearly impossible.

In NS3, we used the wifi module `YansWifiChannel`. We performed evaluations using the original CCNx 0.8.2 implementation and compared it to our Dynamic Unicast implementation.

In order to save time and parallelize the evaluations, we extended NS3-DCE such that it can run on UBELIX [14].

> UBELIX is a Linux High Performance Computing Cluster used for computational extensive tasks.

Evaluations on UBELIX enabled us to perform many different runs in parallel to get statistically significant results. To process and analyze the resulting huge amount of log data, an evaluation framework has been written in Python and Bash .

4.2 Evaluation Setup

In this section the evaluation parameters and metrics will be explained. In addition, global parameters which are used in all evaluation scenarios, are listed in this section and will no longer be mentioned in each individual result section.

4.2.1 Parameter Description

This section describes all parameters that were used in this thesis. They are divided into topology (containing information about nodes' position and movement), simulation (simulation duration,

simulation skip time and number of runs for each scenario), network (wireless transmission parameters) and CCNx, including the two forwarding strategies for Dynamic Unicast, i.e., `SFF` (Dynamic Unicast single face forwarding) and `PFF` (Dynamic Unicast parallel faces forwarding).

Parameter	**Description**
Topology	
Simulation Area	Dimension of the simulation area.
Requester Nodes	Number of nodes interested in receiving content.
Source Nodes	Number of nodes offering content.
Mobile Nodes	Number of nodes providing forwarder capabilities
Source Position	Position of the source node in the area
Requester Position	Position of the requester node in the area.
Node Speed	Speed of the mobile nodes.
Pause Time	Pause time of mobile nodes.
Simulation	
Simulation Length	(Simulation Skip Time + Expire Time).
Simulation Skip Time	Number of seconds to run the mobility model before events will be started.
Run Number	Number of runs per scenario.
Network	
Wireless Standard	IEEE Wireless Standard
Wireless Channel	Frequency of the used wireless channel.
Log Distance Exponent	Exponent for the calculation of the Log Distance Model
Reference Loss	Loss value for the calculation of the Log Distance Model
Energy Detection Threshold	Minimal signal strength to detect an incoming packet.
CCAMode1 Threshold	Minimal signal strength to detect a CCA packet.
CCNx	
File Size	Size of the transmitted data.
Expire Time	Maximum number of minutes to retrieve data.
Strategy	The used CCNx forwarding strategy, i.e., SFF or PFF
SFF	Dynamic Unicast Single Face Forwarding Strategy
PFF	Dynamic Unicast Parallel Faces Forwarding Strategy
Reap Factor	Frequency for checking validation of face entries
Time to Stale	Time that the Data message is considered as valid in cache

Table 4.1: Parameter Description

4.2.2 Global Parameters

This section shows the global parameters. These values are the same for all evaluations unless otherwise specified. They are divided into `Network` (NS3 wifi radio settings), `Simulation` (number of runs and skip time), `CCNx` specific parameters and `Routing` parameters.

To enable multi-hop routing, it is necessary to configure at least two broadcast faces, since forwarding Interests on the same face from where they came from is not possible. Thus, two broadcast ports are necessary to create the two faces. To avoid IGMP group membership messages [15] by using the CCNx multicast address `224.0.23.170`, the all-groups multicast address `224.0.0.1` is used instead. As requester application, `ccnacat_resume` (see subsection 3.6 for more information) is used, which is advantageous in mobile environments with frequent disruptions due to the resume capability. With the selected wireless network parameters in Table 4.2, the transmission range corresponds to approximately 130m. However, closer nodes can communicate with higher data rates (rate adaptation) due to a higher signal-to-noise-ratio (SNR).

Parameter	**Description**
Network	
Wireless Standard	IEEE 802.11g
Wireless Channel	2.4GHz
Log Distance Exponent	3.0
Reference Loss	40.0 dB
Energy Detection Threshold	-86.0 dBm
CCAMode1 Threshold	-90.0 dBm
Simulation	
Simulation Skip Time	7200s (2h)
Number of Runs	100
CCNx	
Reap Factor	2
Time to Stale	300s
Expire Time	10800s (3h)
Routing	
Broadcast Address	224.0.0.1
Broadcast Ports	96595, 96596
Transfer Application	ccnacat_resume
Pipeline Size	4 (static scenarios) / 1 (mobile scenarios)

Table 4.2: Global Parameters

4.2.3 Metrics

Table 4.3 shows the evaluation metrics. The `Retrieval Time` denotes the time to retrieve a file of specific size from the content source. The metric `Status` shows whether a request succeeded or expired. In mobile opportunistic networks, it is not always sure that data arrives in a certain time interval i.e., even if the expire time is long, it is possible that requesters may receive only a few messages or nothing at all depending on the topology and node density. Thus, an expire time ensures that the simulation does not run indefinitely long. If many runs expire (incomplete results), the evaluation results may not be representative because only the best (fastest runs) are considered. However, we selected a large expire time in our evaluations, i.e., 3h for a 5MB file, thus, many expirations indicate very bad performance.

Metric	**Description**
Time	
Retrieval Time	Retrieval time for a file in seconds
Status	
Transfer Status	Status of the transfer: done or expired.
Message Related	
Interest_from	Number of incoming Interests on a specific face.
Interest_to	Number of outgoing Interests on a specific face.
Interest_expiry	Number of expired Interests on a specific face.
Content_from	Number of incoming Content Objects on a specific face.
Content_to	Number of outgoing Content Objects on a specific face.
Content_dup	Number of duplicated incoming Content Objects on a specific face.

Table 4.3: General Metrics

4.3 Multi-hop in a line topology

In this scenario, we evaluate multi-hop communication via broadcast, Dynamic Unicast and unicast. The requester, intermediate forwarders and the content source are set on a line. Only direct neighbors can see each other, such that multi-hop forwarding is required from a requester to the content source. The main goal of this scenario is to compare the throughput of Dynamic Unicast compared to unicast and broadcast in an ideal multi-hop environment.

4.3.1 Scenario Configuration

Figure 4.1 illustrates the topology. Each node only sees its neighbor nodes. The content source is placed at one end and the requester at the other end.

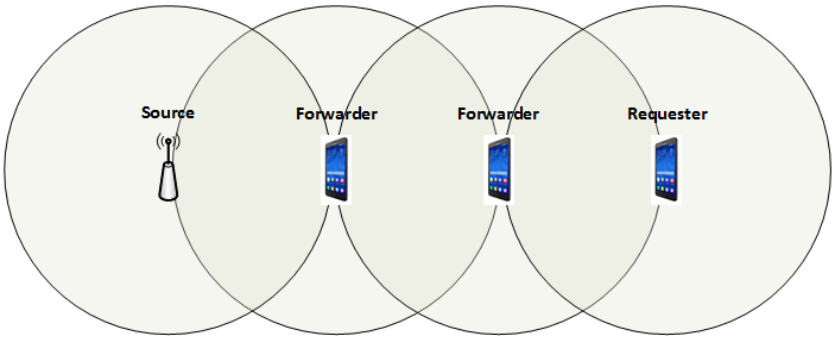


Figure 4.1: Multi-hop in a line - Topology

For the unicast test, the routes are setup statically at the beginning. Each requester has a configured unicast face to the next neighbor towards the content source. The face is configured with the the static prefix `ccnx:/ccnx.org/5MB`. For broadcast or Dynamic Unicast, no path (neighbor) configuration is required, i.e., only two broadcast faces either with the specified static prefix of the content (broadcast) or two broadcast faces with an unspecified prefix (`ccnx:/dummy`) for Dynamic Unicast. For Dynamic Unicast no prefix configuration is required because a multi-hop `pass-through` is used to configure the correct path.

Table 4.4 lists the scenario specific parameters. The `Topology` specifies parameters such as distance to the neighbor node, number of requesters, content sources and forwarder nodes. Neighboring nodes have a distance of 75m, i.e., only direct neighbors see each other due to a transmission range of 130m. The number of forwarding nodes specifies the hop distance between requester and content source, i.e., with 0 forwarders, they are in one-hop (direct) transmission range while with 14 forwarders, they are 15 hops apart. The `Simulation` part shows the NS3-DCE specific parameters such as simulation length, which was set to 10800 seconds (3 hours). The scenario is static, thus, no mobility. Finally, the communication specific parameters are listed under CCNx. In this scenario a 5MB file is transmitted. Since there is always only one forwarding node, we only evaluated the `SFF` strategy, since the `PFF` strategy would produce the same results.

Parameter	Description
Topology	
Distance to Neighbor Node	75m
Requesters Distance to Source	75m, 150m, ..., 1125m
Requester Nodes	1
Source Nodes	1
Forwarder Nodes	0, 2, 4, 6, 8, 10, 12, 14
Simulation	
Mobility Model	Static
Simulation Length	10800s (3h)
CCNx	
File Size	5MB (1281 content objects)
Scenarios	Broadcast, Unicast, `SFF`
Pipeline Size	1, 4

Table 4.4: Multi-hop in a line - Parameters

4.3.2 Scenario with pipeline size 4

This Section evaluates a scenario with a pipeline size of 4, the default pipeline size of CCNx.

Content Retrieval Time

Figure 4.2 shows content retrieval times for a 5MB file by a requester from a content source multiple hops away. The x-axis displays the number of hops, the y-axis shows the transfer time in seconds on a logarithmic scale. Each graph stands for a transmission mode: blue for the `SFF`, green for unicast and purple for broadcast. Figure 4.2 shows that `SSF` is almost as fast as unicast over multiple hops. There is only a small overhead to configure the routes after an initial broadcast. However, broadcast communication results in longer content retrieval times (top line in Figure 4.2).

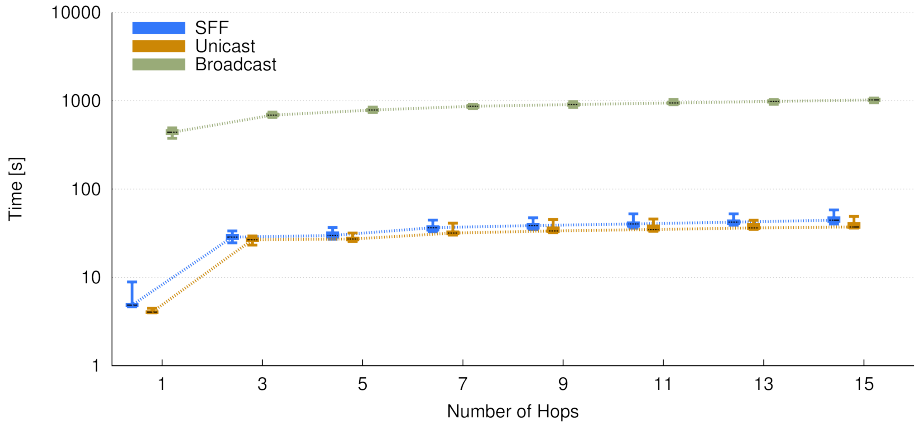


Figure 4.2: Multi-hop in a line - Content retrieval time for a 5MB file.

To see more detailed results of the graph, Table 4.5 shows the median transfer times for multiple hops. Table 4.6 shows the overhead compared of broadcast and `SFF` compared to unicast in percent. We can see that broadcast content retrieval times increase drastically with increasing hop distance while unicast retrieval time increases on a much lower level. `SFF` is, depending on the number of hops, between 7.02% and 20.15% slower than unicast. We can see that broadcast transmission times increase drastically with increasing hop distance while unicast transmissions increase on a much lower level.

The reason for this is twofold. First, broadcast transmissions in `CCNx` are delayed by a random delay to avoid duplicate transmissions. Second, broadcast data rates are slower than for unicast data rates. Thus, the more retransmissions are performed, i.e., the more hops, the more significant this effect becomes. Although we use here the minimal delay of 10ms per transmission, the delay accumulates over multiple hops.

Hops	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
Unicast [s]	4.04	26.72	27.17	31.82	33.58	34.89	36.35	37.25
SFF [s]	4.86	28.60	29.84	36.55	38.59	40.29	42.30	44.45
Broadcast [s]	440.40	687.55	785.91	866.18	905.99	944.48	983.39	1023.29

Table 4.5: Multi-hop in a line, content retrieval time in seconds.

Hops	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
SFF [%]	20.15	7.02	9.83	14.87	14.91	15.48	16.37	19.33
Broadcast [%]	10787.72	2472.73	2792.77	2622.51	2598.17	2607.28	2605.32	2646.87

Table 4.6: Multi-hop in a line, transfer time in percent relative to unicast.

4.3.3 Scenario with pipeline size 1

For comparison, we did the same scenario as well with pipeline size of 1.

Content Retrieval Time

Figure 4.3 shows content retrieval times for a 5MB file by a requester from a content source multiple hops away. The x-axis displays the number of hops, the y-axis the transfer time in seconds on an logarithmic scale. Each graph stands for a transmission mode: blue for the `SFF`, green for unicast and purple for broadcast. Figure 4.3 shows that Dynamic Unicast is almost as fast as unicast over multiple hops. There is only a small overhead to configure the routes after an initial broadcast. However, broadcast communication results in longer transmission times (top line in Fig. 4.3). To see more detailed results of the graph, table 4.7 shows the median content retrieval times for multiple hops. Table 4.8 shows the overhead compared to unicast in percent. Dynamic Unicast is, depending on the number of hops, between 4.37% and 6.72% slower than unicast.

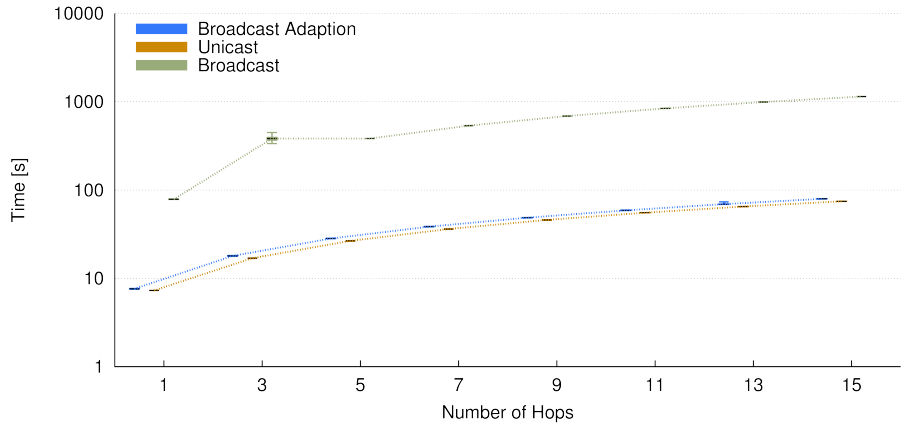


Figure 4.3: Multi-hop in a line - Content retrieval time for a 5MB file.

Hops	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
Unicast [s]	7.31	16.95	26.60	36.24	45.89	55.45	64.99	74.53
SFF [s]	7.63	17.98	28.30	38.50	48.69	58.90	69.28	79.55
Broadcast [s]	78.70	386.34	383.01	535.10	687.43	839.65	991.90	1143.95

Table 4.7: Multi-hop Line, content retrieval time in seconds.

Hops	1	3	5	7	9	11	13	15
SFF [%]	4.37	6.08	6.41	6.23	6.11	6.22	6.61	6.72
Broadcast [%]	976.59	2179.42	1340.10	1376.43	1398.00	1414.19	1426.29	1434.79

Table 4.8: Multi-hop in a line, transfer time in percent relative to unicast.

4.3.4 Summary

In a static scenario, Dynamic Unicast performs approximately as good as unicast (very low overhead). However, Dynamic Unicast has the advantage that no initial route configuration is necessary. Evaluations have shown that the switch form initial broadcast requests to unicast requests happens fast.

In this static scenario, having 15 hops, `SFF` performs with pipeline size 4 78.97% better than with pipeline size 1.

4.4 Multi-hop communication with static requester and source

In this scenario, a static node (requester) requests content from another static node (content source), which is several hops away. Between requester and content source are multiple mobile nodes (forwarders), which can forward requests from the requester towards the content source. Please note that we only use a static requester and a static content source to ensure multi-hop communication over the same distance. If those nodes would move as well, single-hop communication may be performed in some cases. Although the nodes are static, we do not exploit this information and do not route based on the nodes' location (no geographic routing).

Our evaluation results have shown, that in our mobility scenarios pipeline sizes of 1 perform better than larger pipeline sizes. In the following section, all tests were done with pipeline size 1. Please find the results including a comparison with pipeline size 4 in Appendix 6.1.

Adaptive algorithms to increase/decrease the pipeline size depending on the environment are beyond the scope of this work. However, they may be further investigated in future work. For the remaining mobile evaluations, we will stick to a pipeline size of 1.

4.4.1 Topology and Parameters

The evaluation topology is shown in Figure 4.4. The requester is in the bottom left corner and the content source in the upper right corner. Due to the mobility of the nodes in-between, the hop distance between requester and content source can vary (depending on the selected forwarding nodes) although the absolute distance does not.

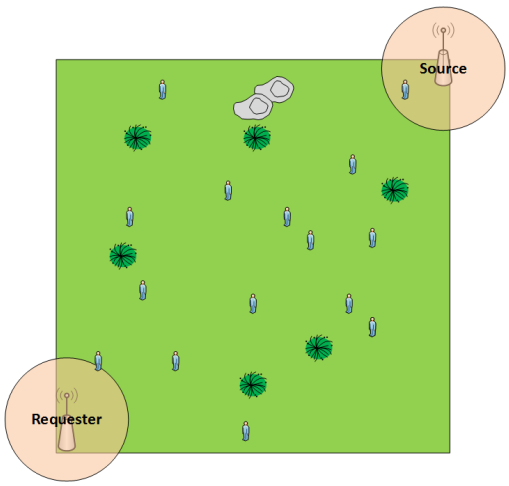


Figure 4.4: Multi-hop communication with static requester and source - Topology, between source and requester are nodes which moving around.

Table 4.9 lists the parameters for this scenario. `Topology` describes the playground settings. There are 50 mobile nodes (forwarders) in the network, which move according to the Random

Waypoint mobility model with node speeds of 1.2m/s (pedestrian speed) or 14m/s (vehicular speed). The distance between requester and content source is varied between 100m (1 hop) and 1000m (8 hops). The transmitted file size is 5MB, which corresponds to 1281 segments. Furthermore, we evaluate the influence of multiple requesters, i.e. 2 up to 32 concurrent requesters compared to 1 requester. In addition, we evaluate the influence of mobility, i.e., mobile nodes with different node pauses compared to static nodes. The `CCNx` Section lists the CCN parameters such as communication types (broadcast, `SFF` or `PFF`), as well as the expiration time. We used an expiration time of 3h, which means that if the content cannot be retrieved within this time, it is considered as expired.

Parameter	**Description**
Topology	
Requester Distance to Source	100m, 250m, 500m, 750m, 1000m
Source Nodes	1
Number of simultaneous requests	1, 2, 4, 8, 16, 32
Forwarder Nodes	50
Simulation	
Mobility Model	Random Waypoint
Node Speed	1.2m/s / 14m/s
Node Pause	0s, 150s, 300s, 1000s, 3600s, Static
Simulation Length	21600s (6h)
Simulation Skip Time	7200s (3h)
CCNx	
File Size	5MB (1281 content objects)
Communication Types	Broadcast, `SFF` and `PFF`
Expire Time	180m (3h)

Table 4.9: Multi-hop communication with static requester and source - Parameters

4.4.2 Pedestrian mobility with single request

In this scenario, mobile nodes move with a pedestrian mobility and a speed of 1.2m/s. The Interests from a requester need to be forwarded by mobile nodes forming opportunistic routes to the content source. The main goal of this evaluation is to compare Dynamic Unicast with broadcast communication and see whether it can work in mobile environments. We evaluate both mechanisms in terms of content retrieval time and message overhead.

Reliability

In the evaluations we set an expiration time of 3h. This means that all content needs to be retrieved within 3h, otherwise the content download expires and the download is aborted. In Figure 4.5, we show the number of successful and expired runs (out of 100 runs) for Dynamic Unicast and broadcast.

We can see that already for a distance of 750m, broadcast file transfer expires before completion in 21% of the cases, which means that the transmission time was longer than 3h. Whereas Dynamic Unicast succeeds in 100% of the cases. For a distance of 1000m, broadcast is barely successful, i.e., only 2 out of 100 runs can be finished. The retrieved partial file sizes in the expired runs are between 250KB and 1.2MB, i.e., less than a fourth of the complete file. Thus, it takes considerably more time to complete the file transfer over this distance. If expire time would be increased to, e.g., 50h, the success rate of broadcast and Dynamic Unicast would increase. However, since such high content retrieval times are not reasonable for a 5MB file we classify these downloads as failed. For the 1000m distance, `SFF` is successful in 53% and `PFF` in 56% of the runs, which is slightly higher due to the exploitation of multiple paths.

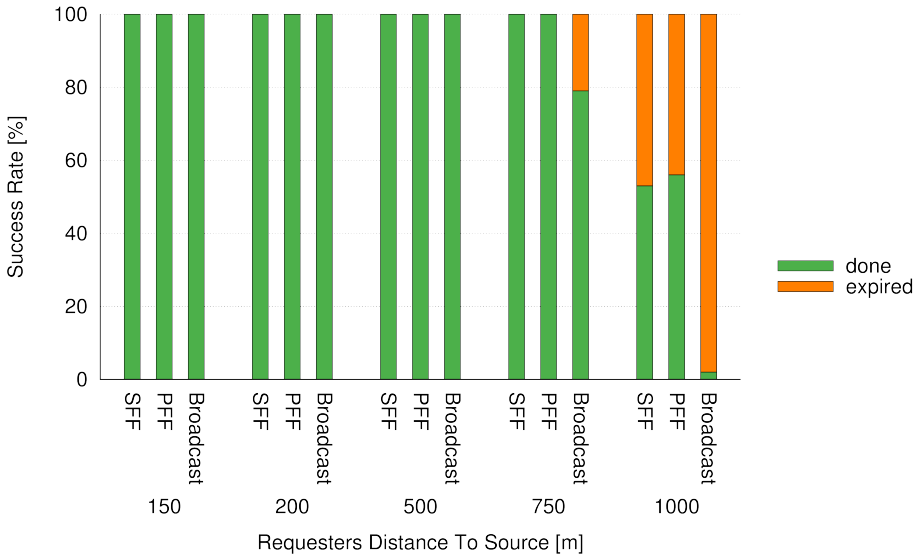


Figure 4.5: Multi-hop communication with static requester and source - Successes for pedestrian mobility with 1.2m/s node speed and 0s node pause.

Content Retrieval Time

Figure 4.6 illustrates the content retrieval times (box plots) of a 5MB file for different distances. The x-axis displays the distance in meters between requester and source. The y-axis displays the retrieval time in seconds on an logarithmic scale. The wireless transmission range of the nodes is 130m (see Table 4.2). The number of hops can be calculated by $\lceil Distance/(transmission \quad range) \rceil$, which needs to be rounded up to the next integer value. Thus, for a distance of 100m, communication can be performed via one-hop, while for a distance of 1000m 8 hops are required in the best case (minimum).

Figure 4.6 shows that Dynamic Unicast (both strategies) can result in significantly shorter content retrieval times. However, the longer the distance between requester and content source becomes, the lower is the difference between both approaches and the closer is the performance to broadcast. For multi-hop communication, `SFF` is between 46.66 (for 250m) and 1.52 (for 1000m) times faster than broadcast. We can also observe that `SFF` is always better than `PFF`, i.e., for 100m it is 1.71 times faster but with increasing distance the differences decrease and with 1000m, `SFF` is only 4.3% faster than `PFF`. Furthermore, we can see that the min-max variation and the inter quartile ranges increase with Dynamic Unicast for increasing distance due to lower node density (less stable paths and more disruptions) since we keep the same number of mobile nodes in the network. Up to 500m (corresponding to 4 hops in the best case), the performance of Dynamic Unicast is very good, but it decreases drastically for longer distances. There are two main reasons for the performance degradation:

1. **Chosen Route** - Dynamic Unicast builds unicast routes to the content source. In case of mobility and long distances, there is a high probability that the route breaks somewhere and a new path needs to be established.

2. **Node Density** - Since the number of mobile nodes in the network is kept static (50 nodes) there are fewer potential forwarders with increasing distance. Thus, if a route breaks, fewer alternatives are available. Due to fewer neighbor nodes, the performance of `SFF` and `PFF` becomes similar (750m and 1000m in Figure 4.6). To confirm this observation, we have also evaluated the 750m distance with 113 nodes to keep the same node density as in the 500m level. The performance results were significantly better (see Appendix 6.2 for more details on this evaluation).

Special attention needs to be given to the 1000m distance. As shown before, at this distance, only very few transmissions (2 out of 100 runs) can finish with broadcast within the expire time of 3h. This means that the results shown in Figure 4.6 show only the fastest runs, while most runs would take significantly more time. With `SFF` and `PFF`, the retrieval times are much lower than with broadcast, thus, much more evaluations can be finished within expiration time.

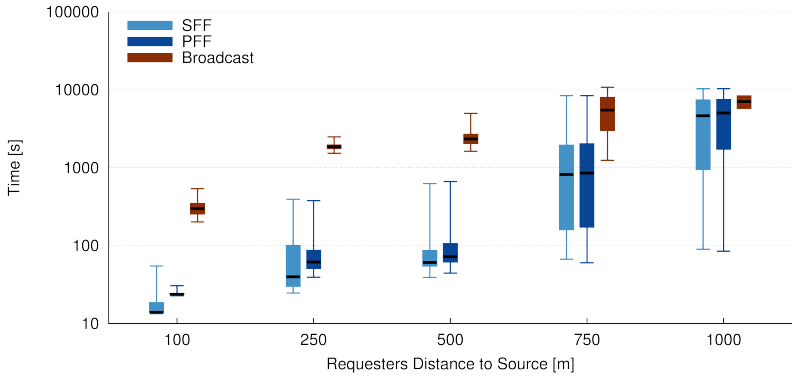


Figure 4.6: Content retrieval time - 50 mobile nodes and distances from 100m to 1000m.

Number of Messages

In this subsection, we evaluate the number of transmitted Interest and Data messages when using Dynamic Unicast and broadcast. Figure 4.7 shows the number of Interests transmitted by forwarder nodes on average. The x-axis shows the distance between requester and content source. The y-axis denotes the number of transmitted Interests on an logarithmic scale. As expected, we can see that Dynamic Unicast with `SFF` sends on average fewer Interests than with `PFF`. Furthermore, `SFF` and `PFF` send fewer Interests than broadcast. In a single-hop environment (100m), `SFF` performs best, resulting in 62.3% fewer Interest messages than broadcast and the same number of Interest messages than `PFF`. Thus, if the content source is within 1 hop distance, `SFF` and `PFF` performs best in terms of Interests transmissions. For a distance of 500m (4 hops), `SFF` results in 68.62% fewer Interest transmissions than with `PFF` and even 83.07% fewer Interests than broadcast. Thus, in case of multi-hop communication, `PFF` strategy performs better than multi-hop broadcast in terms of Interest transmissions but worse than `SFF`.

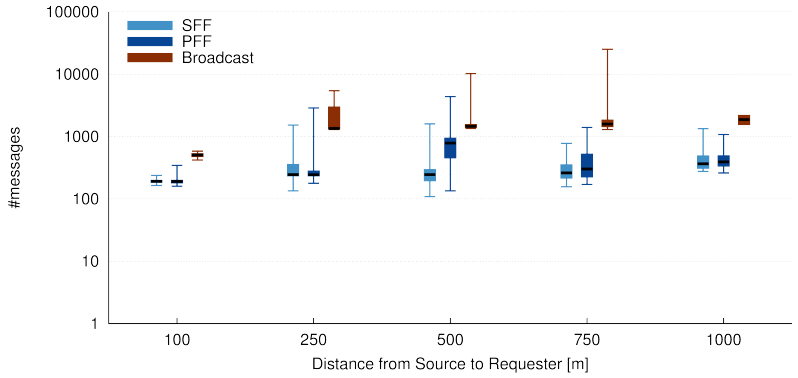
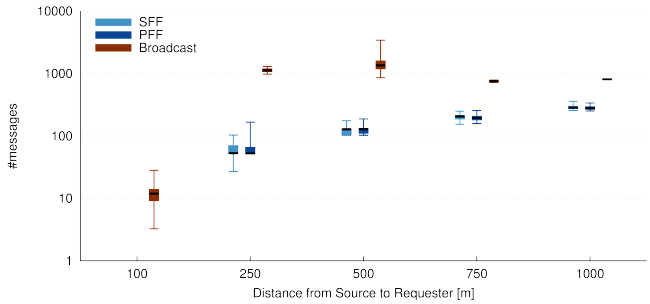


Figure 4.7: Average sent Interest by forwarder nodes with 1.2m/s node speed.

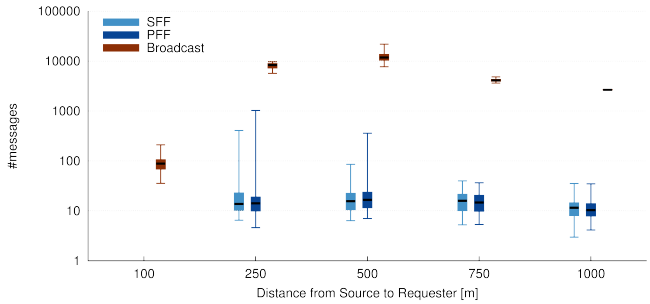
Figure 4.8a illustrates the number of transmitted Data messages from mobile nodes (forwarders). It may look suspicious at first that no Data messages are transmitted by mobile

nodes in case of 100m but keep in mind that requester and content source are within direct transmission range and mobile nodes do not need to forward Interests from the requester. Despite this, mobile nodes still transmit Data messages with broadcast. Because every mobile node has two broadcast faces, they will forward Interests received on one face via the other face. Consequently, if Data is returned on one face, some nodes may forward it via the other face. If Dynamic Unicast is used, a direct path is established such that subsequent requests travel the same path and content returns on the reverse path. For a distance of 500m (4 hops), mobile nodes that use `SFF`, sends on average 128.25 Data messages, while with broadcast, mobile nodes sends on average 1357.64 Data messages (the entire content has 1280 segments). Thus, when using broadcast each forwarder node sends on average 1.06 times the whole content, whereas with Dynamic Unicast, only 9.45% of the content is forwarded by a node on average.

Figure 4.8b shows the number of duplicates received by mobile nodes using Dynamic Unicast and broadcast communication. We can see that Dynamic Unicast results in significantly fewer duplicate content transmissions. For example for a distance of 500m, `SFF` results on average in 16 duplicates and `PFF` on average 17 duplicates while broadcast in 11789 duplicates, i.e., almost 10 times the content. Most duplicates are received for mid-range distances of 250m and 500m and fewer duplicates are received for longer distances. This is due to the mobile node density, i.e., for longer distances, fewer nodes receive and forward Interest and Data messages at the same time via broadcast. Thus, with longer distances (750 meters 1000 meters) the number of duplicates decreases.



(a) Sent Data messages



(b) Duplicate Data messages

Figure 4.8: Data Messages of mobile nodes having 1.2m/s node speed and 0s node pause.

4.4.3 Vehicular node speed with single request

In this Section, we evaluate the same scenario as in the previous section but with a higher node speed. Nodes move with a velocity of 14m/s, which corresponds to 50.4km/h. Thus, the contact times between nodes, i.e., when they can communicate, may become significantly shorter.

Reliability

Figure 4.9 illustrates the reliability of the 14m/s scenario. We can see that all runs were successful up to a distance of 500m. However, for a distance of 750m, 14% of the broadcast transmissions cannot be completed within the expiration time while with Dynamic Unicast all runs have been successfully finished. In general, we can see that for an increasing path length, the success rates decreases significantly because the node density is too low to get a proper path to the source. For example for a distance of 1000m, not a single content download could be finished with broadcast while at least 12% could be finished with Dynamic Unicast. The retrieved partial file sizes for 1000m (expired broadcast runs) are between 150KB and 500KB, i.e., less than a tenth of the complete file. For `SFF` the retrieved partial file sizes are between 12KB and 2MB, thus, it takes considerably more time to complete the file transfer over this distance.

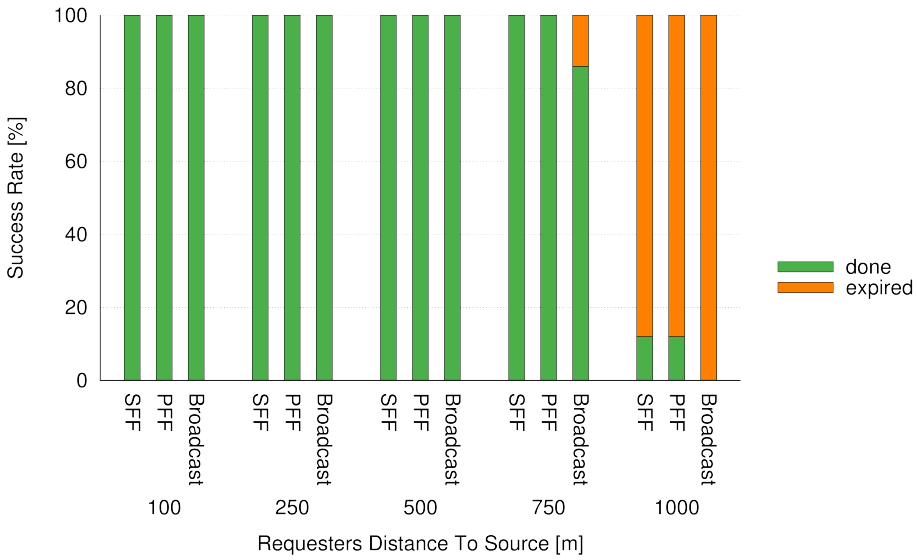


Figure 4.9: Reliability with 14m/s node speed

Content Retrieval Time

Figure 4.10 illustrates the time of a static requester to retrieve a 5MB from a content source if mobile nodes move with a speed of 14m/s. Similar to the pedestrian speed scenario, Dynamic Unicast performs very good up to a distance of 500m. Then, we can observe a significant degradation between 500m and 750m for Dynamic Unicast because the paths to the content sources

are more likely to break. Furthermore, the node density may not be sufficient to guarantee a continuous path between content source and requester at all times. We have repeated evaluations with 750m distance and a higher node density (113 nodes) with significantly better performance (see Appendix 6.2 for more details). Broadcast communication performs for long distances very bad, e.g., for a distance of 1000m not a single run could be completed within the expiration time of 3h.

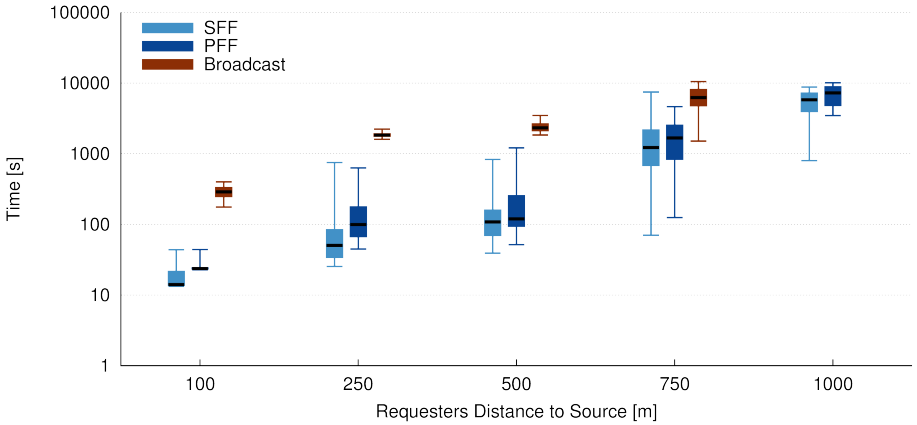


Figure 4.10: Content retrieval time with 14m/s node speed, 0 node pause and 50 mobile nodes

To see the differences between the pedestrian speed (1.2m/s) and vehicular speed (14m/s), we list the median run times for `SFF` (performs better than `PFF`) in Table 4.10 and for broadcast in Table 4.11. The difference is denoted in absolute numbers (seconds) and relative differences (percent). The first column displays the distance between requester and content source.

Table 4.10 confirms that Dynamic Unicast works better with lower speeds because established paths are valid for a longer time. Thus, the time difference increases for increasing path lengths. Up to a distance of 500m it is easy to find an alternative path in case of a path break having pedestrian speeds. This is not the case in the vehicular scenario where paths are valid for a significantly shorter time. The performance also depends on the node density. Up to a distance of 500m, the performance difference between pedestrian and vehicular speeds grows. However, for longer distances of 750m and more, it is also difficult in the pedestrian scenario to get a stable path and alternative paths in case of a path break. Therefore, the performance difference decreases again. Thus, we can say that 50 mobile nodes performing good up to distances of 500m.

Distance [m]	Single 1.2m/s	Single 14m/s	Difference [s]	Difference [%]
100	13.96	14.08	0.12	0.83%
250	39.85	50.53	10.68	26.81%
500	60.93	108.44	37.55	77.97%
750	817.60	1225.92	408.32	49.94%
1000	4645.38	5813.28	1167.91	25.14%

Table 4.10: `SFF` content retrieval time in s, 1.2m/s versus 14m/s.

For broadcast communication the download times do not depend significantly on the mobility speed as Table 4.11 shows. However, broadcast content retrieval times are still significantly higher than with Dynamic Unicast even under high node mobility. For high distances of 1000m, not even a single run finishes with broadcast communication within an expiration time of 3h.

Distance [m]	Broadcast 1.2m/s	Broadcast 14m/s	Difference [s]	Difference [%]
100	297.90	289.28	-8.63	-2.90%
250	1859.23	1836.86	-22.37	-1.20%
500	2336.88	2348.17	11.29	0.48%
750	5473.87	6242.15	768.28	14.04%
1000	7051.58	NA	NA	NA

Table 4.11: Broadcast content retrieval time in s, 1.2m/s versus 14m/s.

Number of Messages

Figure 4.11 shows the number of transmitted Interest messages by mobile nodes. The x-axis denotes the distance between requester and content source in meters, the y-axis denotes the the average number of Interests transmitted by mobile nodes. Having vehicular speed, `SFF` performs best followed by `PFF` and far behind broadcast. The number of transmitted Interests for Dynamic Unicast increases slightly with increasing path length because paths break more quickly.

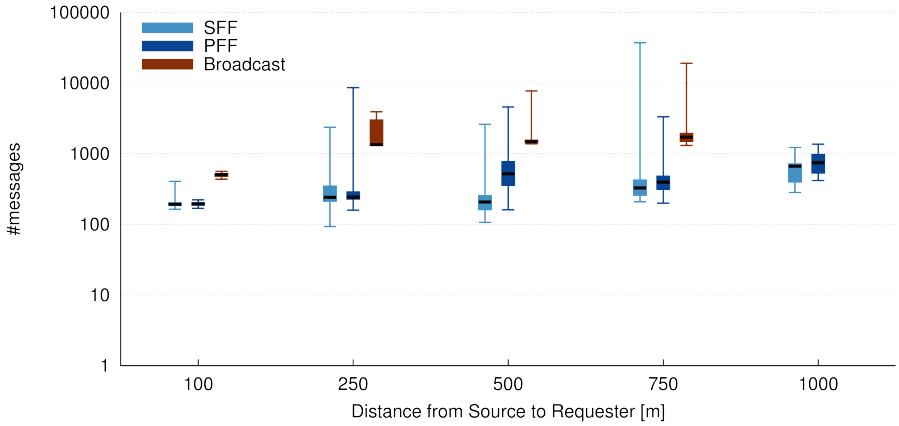


Figure 4.11: Average sent Interest of mobile nodes with 14m/s node speed.

In Figure 4.12, we show the number of transmitted Data messages. To compare the pedestrian with the vehicular scenarios, Table 4.12 lists data messages of pedestrian mobility and vehicular mobility for `SFF`. There is only a small difference between both mobility speeds. Using `SFF` the difference is between 0% (100m) and 6.43% (1000m).

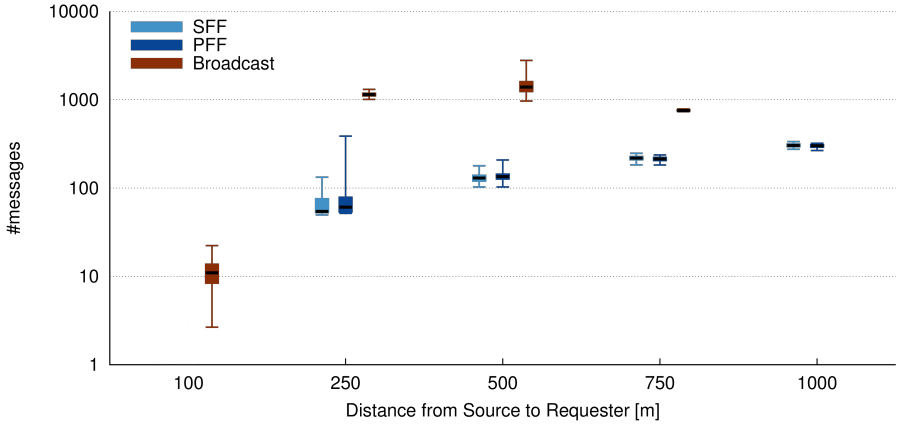


Figure 4.12: 50 Mobile nodes, sent Data messages with vehicular speed an 0s node pause.

Distance [m]	Pedestrian	Vehicular	Diff [msg]	Diff [%]
100	0.00	0.00	0	0.00%
250	52.84	54.37	1.53	2.90%
500	128.25	129.82	1.57	1.22%
750	205.70	218.36	12.66	6.15%
1000	284.92	303.25	18.33	6.43%

Table 4.12: `SFF` transmitted Data messages, pedestrian vs vehicular speed.

4.4.4 Vehicular node speed with different node pauses and single requester

The previous evaluations used a node pause of 0s, i.e., the mobile nodes are moving all the time. However, in a more realistic scenario, some nodes may also take a break from time to time and do not move. Therefore, in this section, we evaluate the influence of different node pause times. A high node pause is more close to a static environment where the routes do not change often. Thus, we evaluate node pauses of 0s, 150s, 300s, 1000s, 3600s as well as a static configuration where the nodes are randomly placed in the simulation area. A node pause of 3600s means that a mobile node waits occasionally between 0s and 3600s until it makes the next movement. The distance between requester and content source is set to 500m and the mobile speed is 14m/s.

Content Retrieval Time

Figure 4.13 displays the content retrieval times in seconds (y-axis) for a 5MB file with different node pauses (x-axis). We can see that Dynamic Unicast with `SFF` or `PFF` results in significantly better performance than broadcast in all cases. We can observe that the content retrieval times decrease with longer node pauses (shortest time in the static case).

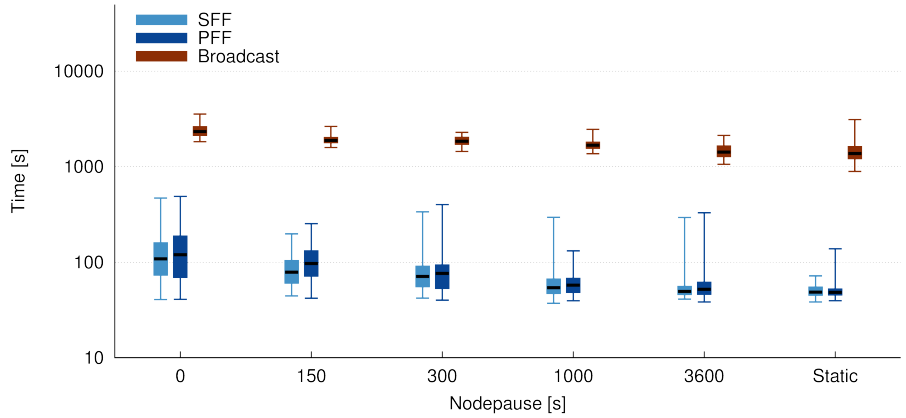


Figure 4.13: Content retrieval time with 14m/s node speed, 500m distance to source and different node pause.

Due to the logarithmic y-axis in 4.13 (broadcast retrieval times are significantly longer), Table 4.13 illustrates the relative differences (percentages) compared to the high mobility case, i.e., a node pause of 0s corresponds to 100%. `SFF` requires 55.18% less time in the static scenario compared to the high mobility scenario (data pause 0s). `PFF` requires 59.68% less time than in the high mobility scenario due to fewer forwarding paths, however, the retrieval time is still higher than with `SFF`.

Broadcast performs slightly better for high mobile speeds compared to `SFF` and `PFF`, however, even broadcast performs worse with short node pauses. In the static case, broadcast requires 41.06% less time compared to the high mobility case. Due to the higher mobility, end-to-end paths may break quicker. This can only be detected after an Interest has expired

(after 4s in our evaluations). Thus, with shorter (dynamic) Interest lifetimes that adapt based on the RTT, the transmission times may be decreased significantly because it takes less time before a retransmission is triggered.

Node pause [s]	0	150	300	1000	3600	Static
SFF	108.44	72.43%	65.41%	49.88%	45.52%	44.82%
PFF	119.80	80.93%	63.62%	47.94%	43.37%	40.32%
Broadcast	2332.10	80.54%	79.10%	71.88%	60.95%	58.94%

Table 4.13: Relative transfer times of with different node pauses. 0s node pause corresponds to 100%.

Number of Messages

Figure 4.14 shows the average number of transmitted Interest messages by mobile nodes. With Dynamic Unicast, the number of transmitted Interests increase only slightly from a static to a high mobile (node pause 0s) scenario. For `SFF`, the Interests increase by 8.02% and for `PFF` they increase by 13.92%. This illustrates that the main reasons for higher retrieval times (see Figure 4.13) are Interest expirations due to lower path stability. For broadcast the number of Interest stays approximately on the same level (significantly higher than `SFF` and `PFF`), because they are flooded to all nodes.

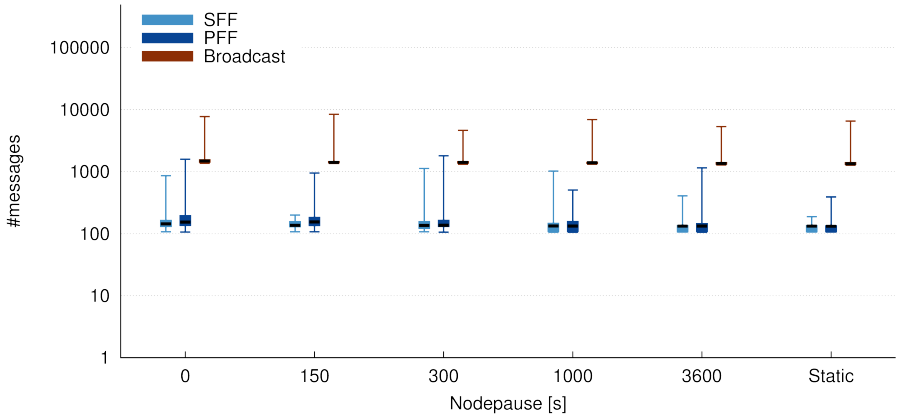
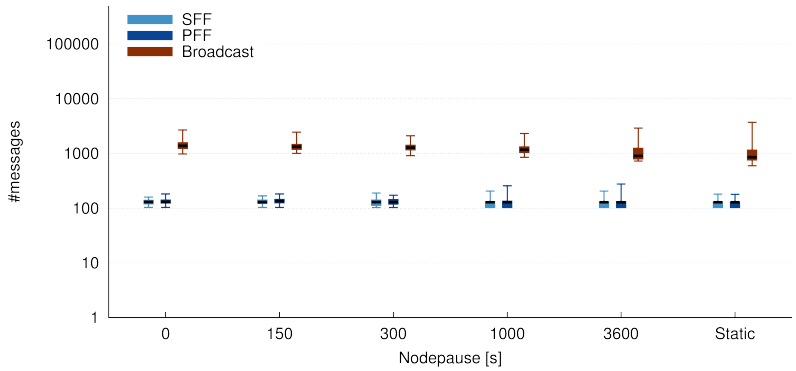


Figure 4.14: Average sent Interest message with different node pauses and 50 mobile nodes.

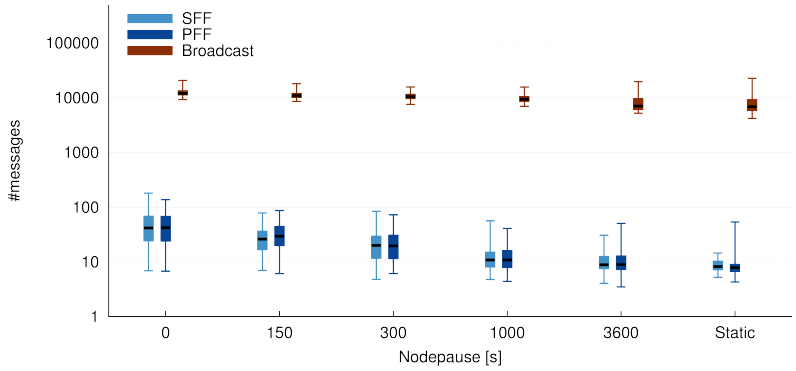
Figure 4.15 shows the transmitted Data messages and received duplicates by mobile nodes. With Dynamic Unicast, the number of transmitted Data messages are kept approximately constant for a static and highly mobile environment because Interest retransmissions can be satisfied from caches. For broadcast, the number of Data messages is 38.15% lower in a static environment compared to the high mobility scenario. This corresponds approximately to the fewer received duplicates (42.69%) in a static environment for broadcast.

In contrast to broadcast, Dynamic Unicast does not flood the network with messages.

Thus, the impact of mobility and Interest retransmissions (broadcast fallback) on received duplicates is higher for Dynamic Unicast. Consequently, compared to a high mobility scenario, 75.43% fewer duplicates are received with `SFF` and 81.42% fewer duplicates with `PFF`. However, even under high mobility, `SFF` and `PFF` result in significantly fewer duplicates than broadcast.



(a) Mobile Nodes - Sent Data messages



(b) Mobile Nodes - Duplicate Data messages

Figure 4.15: Data messages, with different node pause and 50 mobile nodes.

4.4.5 Pedestrian node speed with multiple requester

In this scenario, we increase the number of concurrent static requesters from 1 up to 32 and compare the performance for Dynamic Unicast versus broadcast. The distance between requesters and content source is again set to 500m and there are 50 mobile nodes that are used as forwarders. The requesters are located in a line on the edge of the field. The distance between the requesters is 1m.

Reliability

To see how reliable the evaluation is, Figure 4.16 shows the amount of successes and expiration. All evaluations up to 4 concurrent requesters result in a success rate of 100%. Surprisingly, broadcast does not perform as reliable as expected for 8 and 16 concurrent requests. For broadcast, all requesters receive the data at the same time and request then the subsequent segments. We assume that having 8 requesters, the position of the requester nodes lead to many collisions and, therefore, to a lower delivery rate. For more requesters the delivery rate may increase because even though some requests collide, other requests may be forwarded later and retrieve the content. The delivery rate of Dynamic Unicast is always better or similar than broadcast.

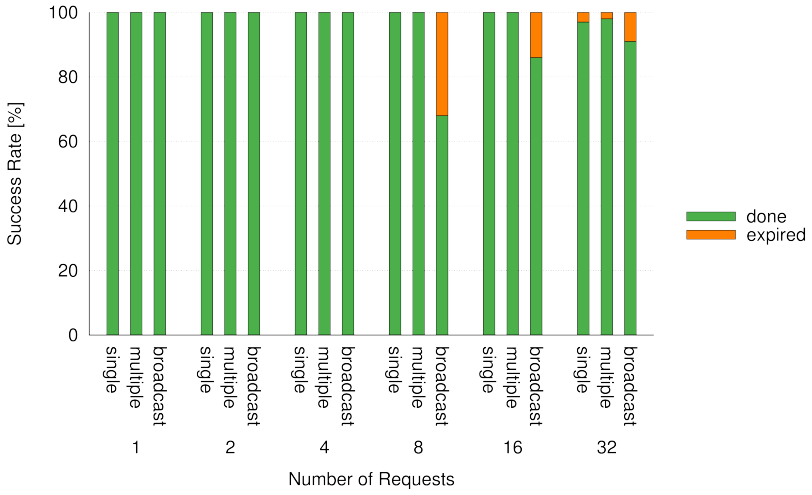


Figure 4.16: Reliability with 1.2m/s node speed and different number of requesters.

Content Retrieval Time

Figure 4.17 illustrates the average content retrieval times for multiple requesters. We can see that Dynamic Unicast performs better than broadcast in terms of content retrieval times up to 32 concurrent requesters. However, for an increasing number of concurrent requesters, Dynamic Unicast converges to broadcast. The average content retrieval times for multiple requesters increases by a factor of 28.92 from 1 to 32 requesters with `SFF`, while for broadcast it increase only by a factor of 1.79. However, broadcast retrieval times are very slow, i.e. it takes more time

than with Dynamic Unicast. With only 1 requester, SFF is on average 38.35 times faster than broadcast, while for 32 requesters it is still 2.39 times faster.

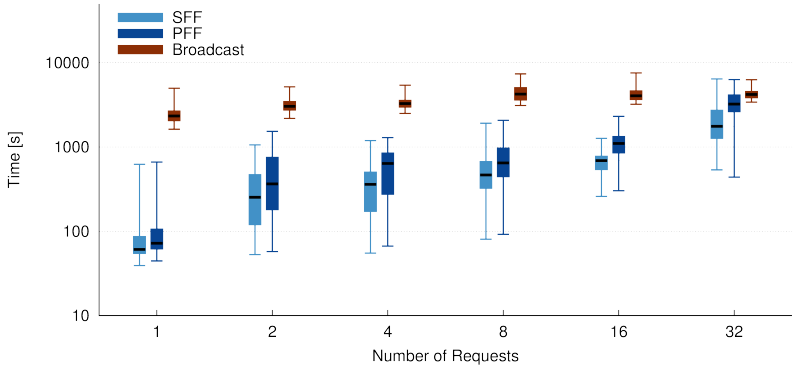


Figure 4.17: Content retrieval time with 1.2m/s node speed, 500m distance and different number of requesters.

Number of Messages

Figure 4.18 illustrated the transmitted Interests by mobile nodes. As expected for Dynamic Unicast, the number of transmitted Interests increase for an increasing number of concurrent requests. However, even for broadcast, the number of Interests increase slightly. PFF sends in average similar or even more Interests than broadcast. This is due to the number of unicast connections that PFF maintains (multiple parallel forwarding faces). For SFF, the number of transmitted Interests increases as well, but they are always below PFF. However, for 32 concurrent requesters with SFF results in slightly more Interest transmissions than broadcast.

For 1 requester, SFF sends on average 246 Interests, which increase to 5392 (21.92 times more) Interests per node for 32 requesters. For broadcast, the transmitted Interests increase from 1453 (1 requester) to 4173 (2.87 times more) for 32 requesters. However, since Interests are rather small messages of 50 bytes, the overhead for 32 requesters is about 61KB.

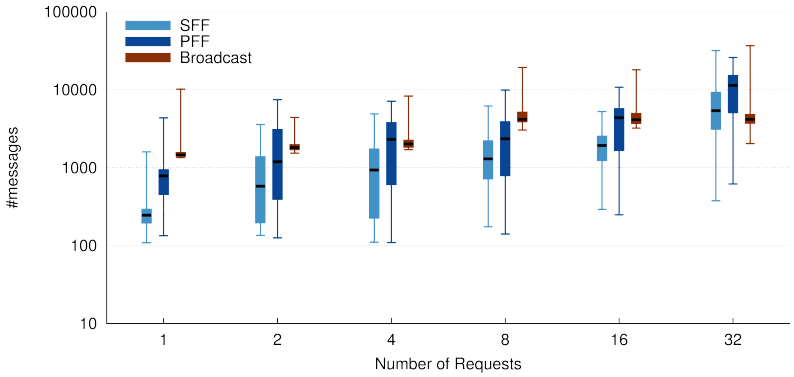


Figure 4.18: Average sent Interest of mobile nodes.

Figure 4.19 shows the average number of transmitted Data messages by mobile nodes. We can see that for an increasing number of concurrent requesters, the number of transmitted Data messages by mobile nodes increases for both Dynamic Unicast and broadcast. However, even for 32 concurrent requesters, the number of transmitted Data messages sent via Dynamic Unicast is still significantly lower than for broadcast, because broadcast floods the content in the network. Furthermore, with Dynamic Unicast, content is still cached by nodes such that each Data message can be retrieved from the nearest cache and does not need to be transmitted over the entire path from the content source. For 1 requester, a mobile node with `SFF` sends on average 128 Data messages, which corresponds to 10% of the total content size, and for 32 requesters 617 Data messages are sent, which corresponds to 48% of the total content size. For Broadcast with 1 requester, each mobile node sends on average 1358 Data messages (106% of the content) and with 32 requesters, this increases even to 5197 Data messages (406% of the total content). We would like to emphasize that Data messages are significantly larger than Interest messages, i.e., they have a payload of 4096 bytes resulting in an overall size of approximately 4500 bytes including headers. Thus, broadcast has a significantly higher communication overhead than Dynamic Unicast, since approximately 21.5MB more Data is transmitted for 32 requesters with broadcast compared to `SFF`.

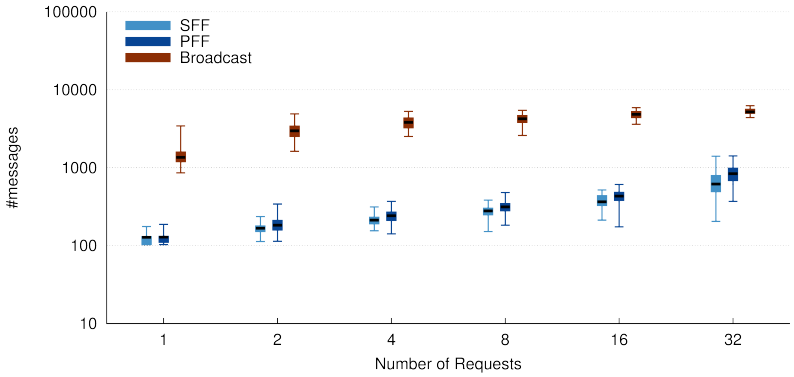


Figure 4.19: Mobile Nodes - transmitted Data messages

4.4.6 Summary

After these extensive evaluations, we can draw the following conclusions. Dynamic Unicast results in significantly faster transmission times than broadcast for 1 or multiple requesters (up to a 32 requesters). The forwarding delays for duplicate suppression and lower data rates of broadcast compared to unicast are the main reasons for the longer transmission times compared to Dynamic Unicast.

Furthermore, Dynamic Unicast results in a much lower message overhead. In the single requester scenario, Dynamic Unicast (`SFF`) sends approximately 83.3% fewer Interest and 91.7% Data messages than broadcast because messages are not flooded in the network. Surprisingly, the message overhead is also slightly lower for multiple requesters. Although Dynamic Unicast can send more Interest messages to find the nearest content copy, significantly fewer Data messages are transmitted. Because content is also cached during unicast communication, the content density increases with multiple multi-hop requesters. As a consequence, fewer Data messages are transmitted compared to broadcast (flooding).

Comparing the two forwarding strategies for Dynamic Unicast, `SFF` is better than `PFF` in almost all scenarios because of fewer Interest transmissions and (slightly) faster content transmissions. Only in high distance low density topologies, `PFF` performs slightly better than `SFF` due to a higher reliability (more redundant paths)

Thus, from all evaluated communication strategies, the best choice in high mobile opportunistic networks is Dynamic Unicast with the `SFF` strategy.

4.5 Content Sources in a Grid Scenario

In this scenario, we consider a static grid of content sources and mobile nodes moving according to the Random Waypoint mobility model. Consider for example a touristic area in the mountains, where cellular coverage may not be sufficient or where most users (tourists) do not have a cellular data subscription, i.e., roaming resulting in high costs. In such a scenario, users can retrieve their information and updates from deployed access points.

Subsection 4.5.1 describes the network topology and scenario in detail and lists the evaluation parameters. In Subsection 4.5.2, we evaluate the communication performance in case of single-hop communications and in Subsection 4.5.3 we show evaluations if multi-hop communication is enabled. In case of node mobility, static routes to content sources are not feasible, thus, we compare Dynamic Unicast only to broadcast and not to unicast. Finally, in Subsection 4.5.4, we compare single-hop to multi-hop and evaluate, which strategy is better, in which case.

4.5.1 Topology and Parameters

Figure 4.20 shows the topology of the scenario and Table 4.9 lists all evaluation parameters. We use a square playground with a side length $l_{playground}$ of 1000m. Depending on the number of content sources $n_{sources}$, the playground is divided into smaller regions and each region has a content source in the middle. The side length l_{region} of these smaller regions is calculated by $l_{region} = l_{playground}/\sqrt{n_{sources}}$. For example, for a grid with 4 content sources (shown in Figure 4.20), each region has a side length of 500m.

Please recall that a node (requester or content source) has a wireless transmission range of approximately 130m in our configuration (See Section `Global Parameters` 4.2.2 for the detailed wireless configuration). Thus, with 16 content sources, the playground is completely covered by content sources, such that direct communication between a requester and a content source is always possible. In all other cases a requester has to walk into the content sources wifi range or has to request data over an opportunistic multi-hop route.

There are multiple (mobile) requesters in the playground. Furthermore, there are mobile nodes, which do not request content and are no content source. These mobile nodes can be used as forwarder nodes (in case of multi-hop communication) or data mules (if overhearing broadcast content transmission). Requesters and mobile nodes are initially distributed randomly in the playground and move according to the Random Waypoint mobility model. Each scenario has always 30 mobile nodes (not actively requesting content), and a varying amount of requester nodes (1 to 30) and content sources (1 to 16). For example in a scenario with 16 content sources, 30 mobile nodes and 30 requesters are in total 76 nodes at the playground.

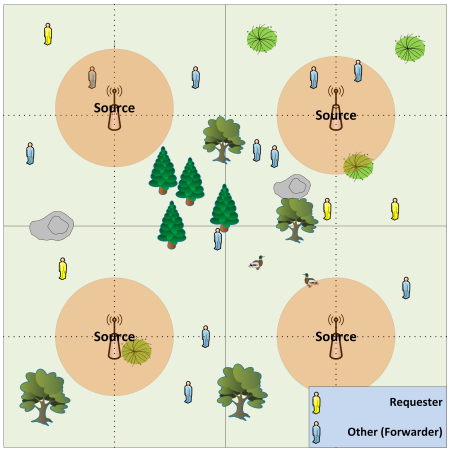


Figure 4.20: Grid Topology: Content sources are placed in a grid in the topology (here with 4 content sources).

Table 4.14 lists the used evaluation parameters. The `Parameter` section lists the different topology combinations of this scenario, i.e., number of mobile requesters, mobile nodes (not requesting content actively) and content sources. The `Simulation` section shows the simulation parameters. All mobile nodes, including the requesters, move according the Random Waypoint mobility model with a node speed of 1.2m/s and a pause time of 0s. The `CCNx` parameters lists the used file size, i.e., 5MB, and the used communication type (broadcast and Dynamic Unicast).

Parameter	**Description**
Topology	
Square length of playground	1000m
Source Nodes	1, 4, 16
Number of simultaneous requests	30
Number of Requesters	according simultaneous requests
Mobile Nodes (not requesters)	30
Simulation	
Mobility Model	Random Waypoint
Node Speed	1.2m/s
Node Pause	0s
CCNx	
File Size	5MB (1281 content objects)
Communication Type	Broadcast, `SFF` and `PFF`

Table 4.14: Grid - Parameters

4.5.2 Single-Hop Only

In this scenario, the communication is only performed via one hop. This can be assured by configuring only one broadcast face per node, i.e., Interests can not be forwarded on another face from where they were received.

Content Retrieval Time

Figure 4.21 shows the content retrieval times of 30 concurrent requesters for a different number of content sources (grid configurations). The x-axis denotes the content retrieval times of 30 mobile requesters. The y-axis denotes the cumulative number of nodes that have received the content at a given time (x-axis). The values are ordered by shapes and colors. Blue colors stand for 16 sources, orange colors for 4 sources and purple colors for 1 source. A square stands for `SFF`, a circle for `PFF` and a triangle for broadcast.

Figure 4.21 shows that Dynamic Unicast (independent of the strategy) performs always better than broadcast, i.e., requesters can retrieve the content much faster. Both forwarding strategies for Dynamic Unicast perform approximately equally well. We can see that the differences between the forwarding strategies are negligible. For higher content source densities, requesters can retrieve the content faster than for low content source densities, because suitable content sources can be found quicker, while for low content source densities, requesters need to first come in range of a content source. For the complete playground coverage, i.e., 16 content sources, Dynamic Unicast results in significantly faster content retrieval times than broadcast. At the time when all requesters have completed content retrieval with Dynamic Unicast, not even 50% of the requesters have completed it with broadcast. With Dynamic Unicast the last of the 30 requester has the content 2.67 times faster than the last requester with broadcast.

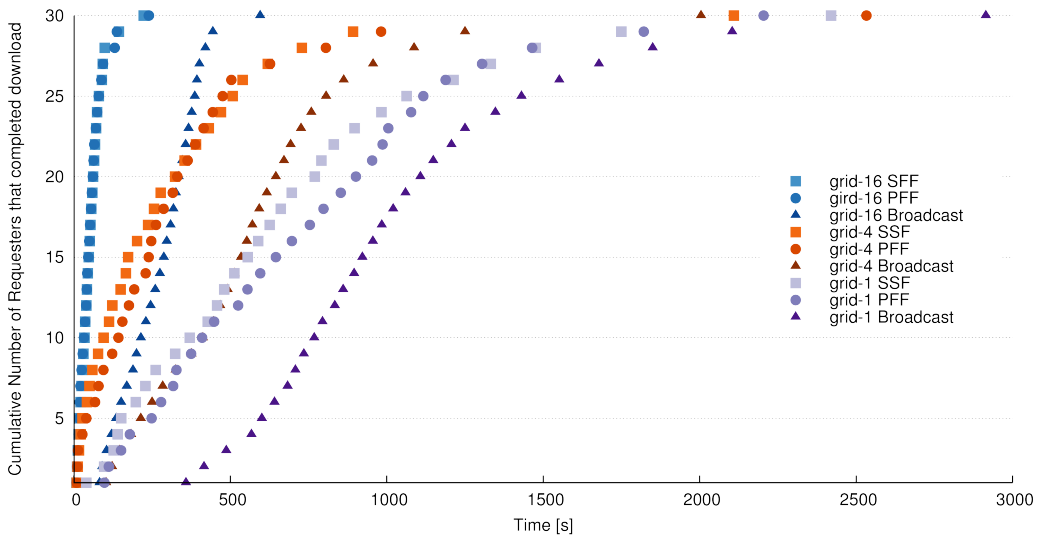


Figure 4.21: Grid Timeline - single-hop, time till each requester has content downloaded complete, with different number of sources.

4.5.3 Multi-Hop

In this subsection, we evaluate content retrieval in the same scenario as above, but with enabled multi-hop communication.

Content Retrieval Time

Figure 4.22 illustrates the cumulative content retrieval times for 30 concurrent requesters retrieving content over multiple hops in different content densities.

For both Dynamic Unicast and broadcast, content retrieval is faster for a higher content density, i.e., content retrieval in a grid with 16 sources is faster than with 4 sources and faster than with 1 source. Dynamic Unicast performs always better than broadcast. For only 1 source, the last requester receives the content 1.64 times faster with `SSF` compared to broadcast. For 4 sources, `SFF` is 1.83 times faster and for 16 sources, it is 6.35 times faster than broadcast. It is also visible that `SFF` performs better than `PFF` in this scenario. This is due to the lower network load, i.e., fewer transmitted Interest messages (see next subsection). Fewer Interest messages means fewer control traffic (to retrieve content), which means that the medium can be better used for Data transmissions. Therefore, a single path with `SFF` and broadcast as fallback performs better.

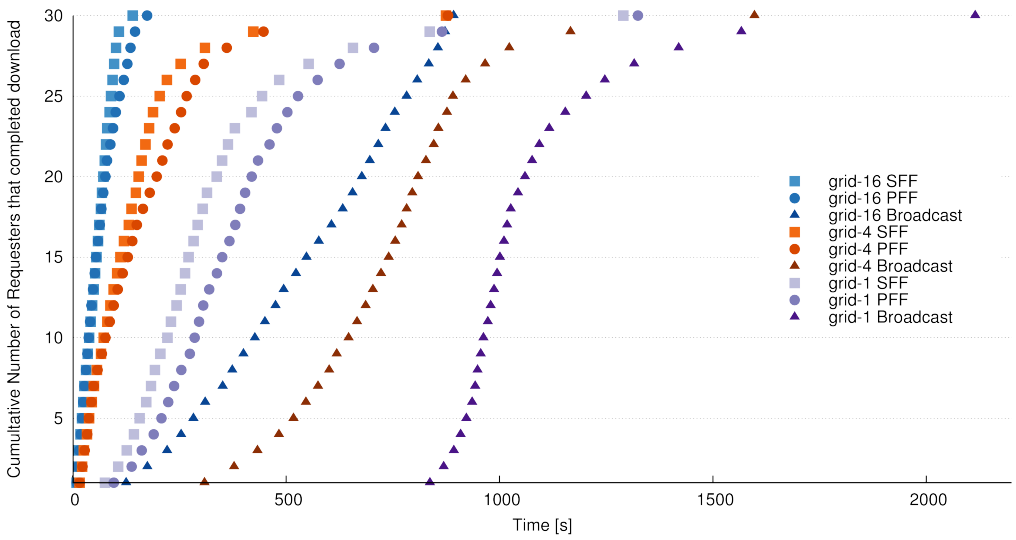


Figure 4.22: GRID Timeline - multi-hop, time till each requester has content downloaded complete, with different number of sources.

The same evaluation has also been performed for a grid with 36 nodes (more nodes than required to cover the area). However, since the content retrieval times did not differ from a grid with 16 nodes, they were not shown in Figure 4.22 (overlapping figures).

Number of Messages

Please note that the values shown in this Subsection are average values over all requester or forwarder nodes of a run. The box plots show min-25-quantil-median-75quantil-max of the average values from 100 runs

Figure 4.23 shows the number of transmitted Interest messages by forwarder nodes, i.e., mobile nodes which do not request any content by themselves. The x-axis shows the number of sources, the y-axis denotes number of transmitted Interest messages. We can see that even for 30 concurrent requesters, `SFF` results in significantly fewer Interest transmissions than with broadcast, because only a few Interests are flooded in the entire network. For Dynamic Unicast, only the first Interest is flooded to setup the route, and the following Interests are transmitted on the path to the content source until the route breaks and a new flooding is required. With an increasing number of sources, the average number of transmitted Interest messages by `SFF` decreases slightly. Since sources become closer, fewer forwarders are involved in multi-hop communication. For example, for 1 content source, forwarders transmit on average 3.22 times more Interests with broadcast than with `SFF`. For 4 content sources, broadcast results in 4.32 times more Interests, for 16 sources in 4.8 times more Interests and for 36 sources in 5.07 times more Interests than `SFF`.

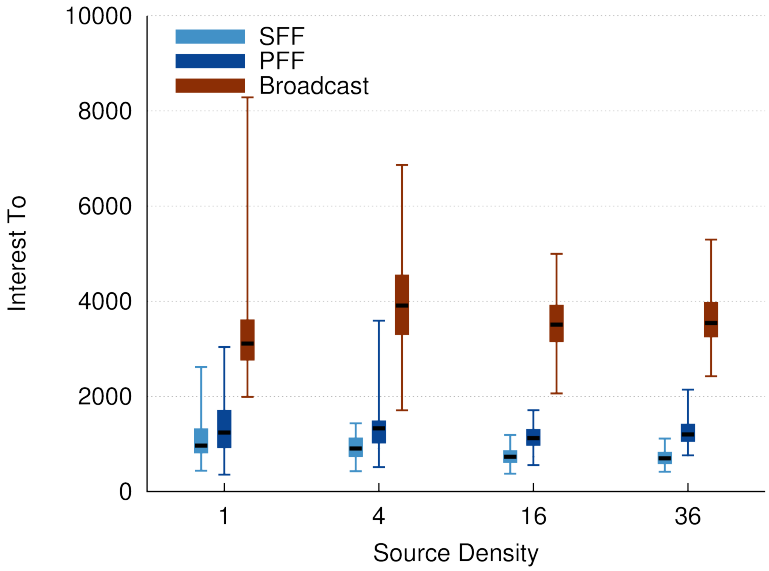


Figure 4.23: Average sent Interests by the 30 mobile forwarder nodes.

Figure 4.24 shows the average number of transmitted Data messages by forwarder nodes. Both SFF and PFF perform better than broadcast. For one content source, forwarder nodes send 203% more Data messages with broadcast than with SFF. For higher content source densities, forwarders with SFF send fewer content messages because the path length decreases. However, for broadcast the number of transmitted content messages does not decrease significantly because they are flooded in the network (following the Interest messages). Consequently, for a source density of 36 content sources, broadcast sends 759.53% more Data messages than SFF.

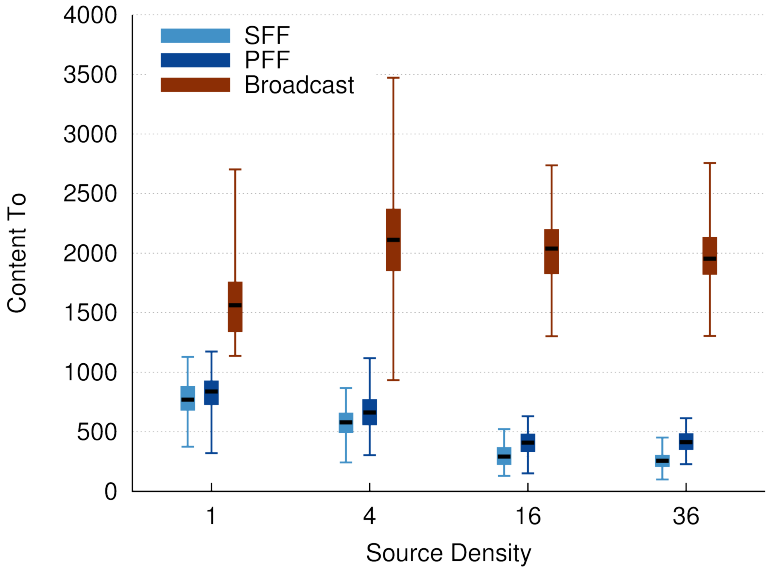


Figure 4.24: Average sent Data messages for different source densities.

Figure 4.25 illustrates the average transmitted Data message by content sources. We can see that for an increasing number of content sources, the number of Data messages that each content source needs to transmit decreases. Surprisingly, Dynamic Unicast performs better than broadcast, i.e., fewer Data messages need to be transmitted despite unicast connections. For only one content source and 30 requesters, the content source sends on average 285.71% of the entire content with SFF and 381.42% with broadcast. For 36 content sources, each content source sends on average 70.26% with SFF and 225.44% with broadcast. This is due to the fact that broadcast requests address multiple content sources at once, resulting in more content transmissions and duplicates, while unicast requests address only a specific source.

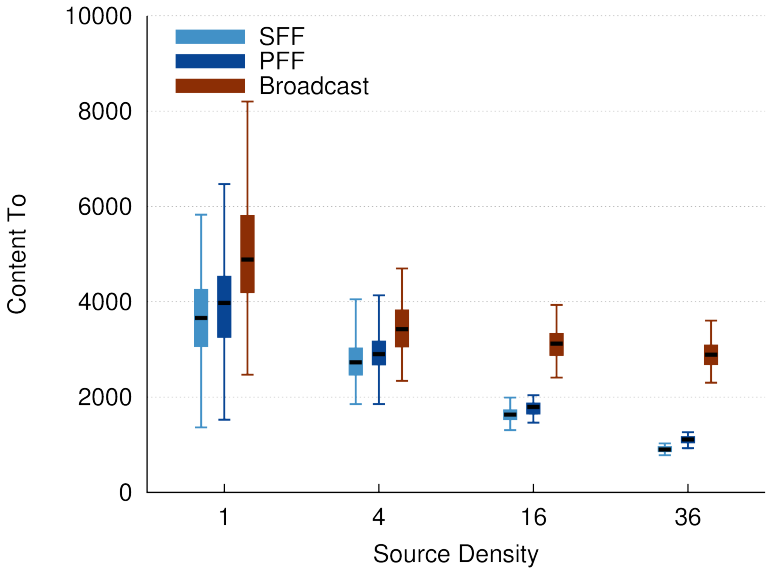


Figure 4.25: Average sent Data messages sent by source nodes.

To conclude this section we can state that `SFF` is the most efficient way for multi-hop communication in a grid topology. Broadcast floods the network with messages, which makes it inefficient and may overflow it. `PFF` produces more messages and results in slower content retrieval times than `SFF`.

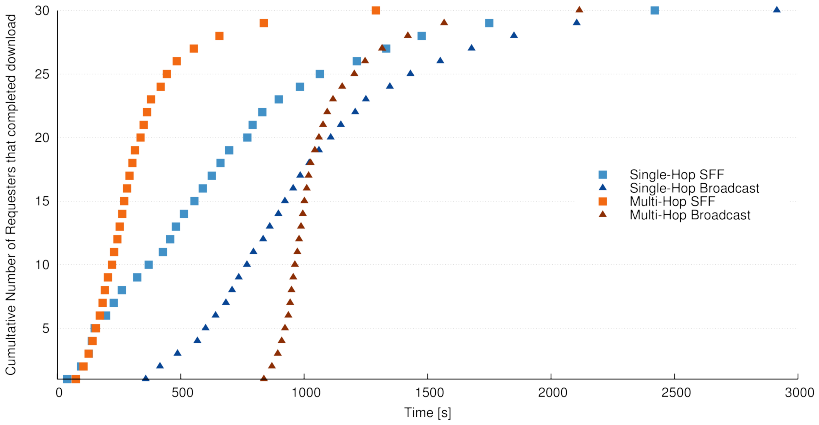
4.5.4 Single- vs Multi-hop

In this section, we directly compare multi-hop and single-hop communication in the scenarios mentioned above. Figure 4.26 shows the comparison for 1, 4 and 16 sources. To get a better overview we ignore `PFF`, which performed in all evaluations equal or slightly worse than `SFF`.

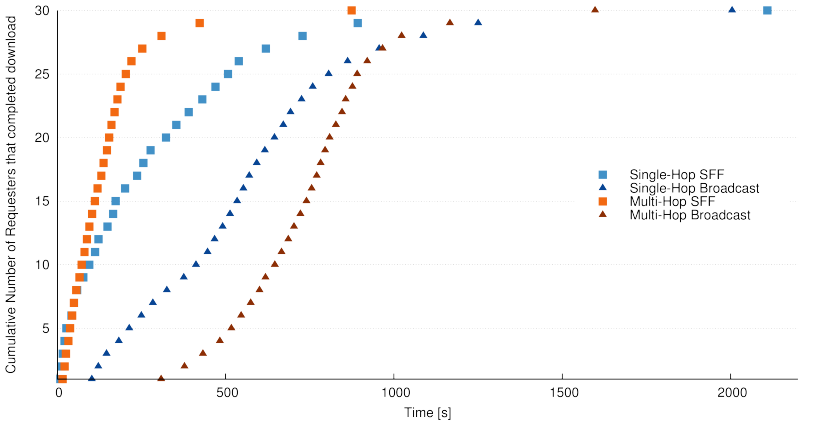
4.5.5 Content Retrieval Time

For only one content source (Figure 4.26a), it is clearly visible that multi-hop communication is superior to single hop. This is because in a multi-hop scenario, a requester can find a content source even if it is multiple hops away and does not need to meet the content source physically, i.e., be in direct communication range. In the single hop scenario a requester has to walk into the content source's wireless transmission range, which may take some time for a pedestrian speed of 1.2m/s. For 4 content sources (Figure 4.26b) Dynamic Unicast via multiple hops is still more efficient than with single-hop communication.

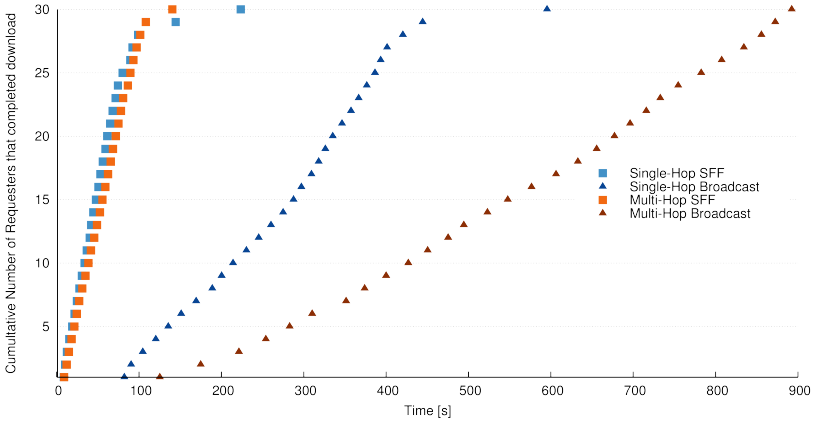
However, the performance of multi-hop broadcast communication degrades for an increasing number of content sources because of a higher content availability in the network. Without multi-hop, requests reach only direct neighbor nodes, which is no disadvantage for broadcast. Only for 1 content source, broadcast multi-hop communication is better than single-hop broadcast. For 4 content sources or more, the content can be found much quicker since it is not required to flood the entire network. For 16 content sources (Figure 4.26c), i.e., complete network coverage with content sources, Dynamic Unicast via single-hop is better than multi-hop. Because the entire network is covered by content sources, broadcast multi-hop has no benefit compared to single-hop communication, i.e., only more messages are transmitted.



(a) Timeline 1 source



(b) Timeline 4 sources



(c) Timeline 16 sources

Figure 4.26: Grid Timeline - Single- vs Multi-hop

4.5.6 Summary

In a grid scenario with enabled multi-hop communication, Dynamic Unicast is very efficient, considering content retrieval times and message overhead. If communication is broadcast only, forwarders send on average 5 -10 times more Interest messages than with Dynamic Unicast and 2-7.5 times more Data messages. Comparing the two Dynamic Unicast strategies, `SFF` is more efficient than `PFF` because it results in fewer Interest transmissions and faster content retrieval times. For a low content source density, i.e., no coverage over the entire playground, multi-hop communication performs much better than single-hop because requesters do not need to go in the content sources' wireless transmission range and can request content from far away. However, if the entire playground is covered with content sources, single-hop communication is more efficient due to fewer network traffic.

Overall we can conclude that `SFF` strategy performed better than `PFF` and broadcast in all evaluated scenarios.

4.6 Circle Mobility

The circular mobility model has been developed in [11], where agent performance was compared with broadcast multi-hop performance. The main goal of these evaluations are to check if Dynamic Unicast is beneficial compared to broadcast in this scenario.

4.6.1 Topology and Parameters

The evaluation scenario [11] is described as an everyday situation and is visualized in Figure 4.27. There is a round trip hiking trail to an important spot on a hill. At this important spot a solar powered sensor gathering data (e.g. weather data, web cam snapshot, etc...) is installed. At the start of the trail is the tourist office, which is interested in the sensor data. Unfortunately, the sensor is too far away for a direct wireless connection and a wired connection is also missing. In order to obtain the data anyway, multi-hop communication is required. Since a lot of hikers are on this trail their mobile phones can be used to build opportunistic networks.

To support this scenario, a circular mobility model has been implemented [11] into NS3, where hikers, i.e., nodes, follow a circular path. The nodes (except requester and content source, which are static) are uniformly random distributed on the whole trail and move with velocities within a specified range. From time to time the nodes take a break to enjoy the beautiful view or eat something. This `pause time` is configurable parameter. The hikers moving at speed between 1.0m/s and 1.4m/s

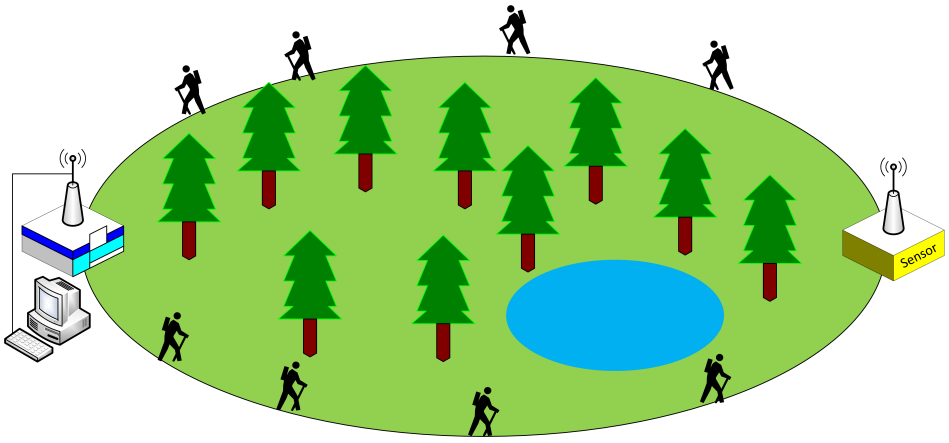


Figure 4.27: Circle Mobility Scenario - Topology

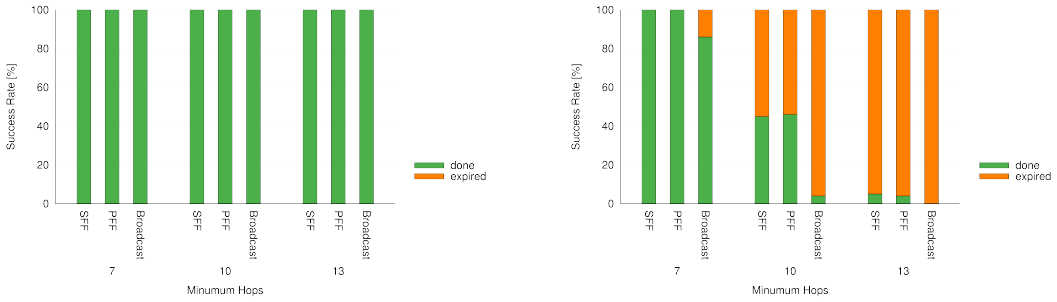
Table 4.15 presents the scenario specific parameters. The `Topology` section shows the circle radii, and the resulting node density. To keep the same node density, the number of mobile nodes has been increased for higher circle radii. The `Simulation` section shows the chosen node speed, which is between 1.2m/s - 1.4m/s with a node pause of 30 seconds. The `CCNx` section shows the framework specific parameters. In this scenario the file size was 1MB. The maximum expire time is set to 180 minutes (3 hours) and all known communication types (broadcast, `SFF` and `PFF`) were tested.

Parameter	Description
Topology	
Circle Radius	250m, 375m, 500m
Requester/Source Distance	785m, 1177.5m, 1570m
Min Number of Hops	7, 10, 13
Source Nodes	1
Number of Requests	1
Node Density	31.42m, 78.54m (average distance to neighbor)
Number of Mobile nodes	20, 30, 40, 50, 75 100 (depends on density)
Simulation	
Mobility Model	Random Waypoint
Node Speed	1.0 - 1.4m/s
Node Pause	30s
Simulation Skip Time	7200s (2h)
Simulation Length	21600s (6h)
CCNx	
File Size	1MB (251 content objects)
Communication Types	Broadcast, `SFF`, `PFF`
Expire Time	180m (3h)

Table 4.15: Circle Mobility - Parameters

4.6.2 Reliability

Having an average distance of 31.42m between nodes (higher node density) there is a overall 100% success rate for all evaluations as shown in Figure 4.28a.



(a) 31.42m between nodes

(b) 78.54m between nodes

Figure 4.28: Circle Mobility Scenario - Reliability

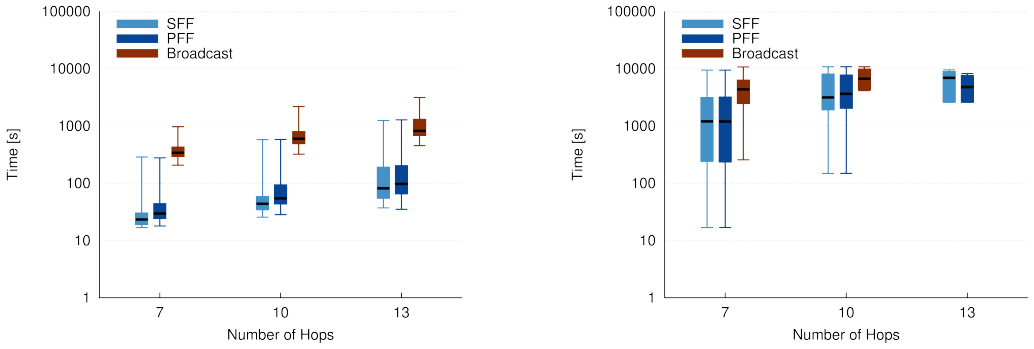
However, for the low density network (on average every 78.54m a node), content retrieval does

not finish in all cases due to disruptions on the path. While for 7 hops the success rate with Dynamic Unicast is still 100%, it is only 86% with broadcast. This means that the file transfer can not be completed in 14% of the cases (lower data rate and longer forwarding delays with broadcast). For 10 hops, the reliability of Dynamic Unicast decreases significantly to 45% (`SFF`) and 46% (`PFF`), however, with broadcast only 4% of the file transfers are successful within the expire time. For 13 hops, not a single run is successful with broadcast while `SFF` results in 5% success rate and `PFF` in 4%.

4.6.3 Content Retrieval Time

Figure 4.29 shows the content retrieval times for a 1MB file by a requester from a content source multiple hops away. The x-axis denotes the minimum number of hops from source to destination. The y-axis illustrates the transfer time on a logarithmic scale. We perform the evaluations for two node densities, i.e., on average a distance of 31.42m or 78.54m between neighboring nodes.

For the high node density (Figure 4.29a), Dynamic Unicast performs significantly better than broadcast. For 7 hops, `SFF` is 14.61 times faster than broadcast and for 13 hops, it is still 10.05 times faster.



(a) 31.42m node density

(b) 78.54m node density

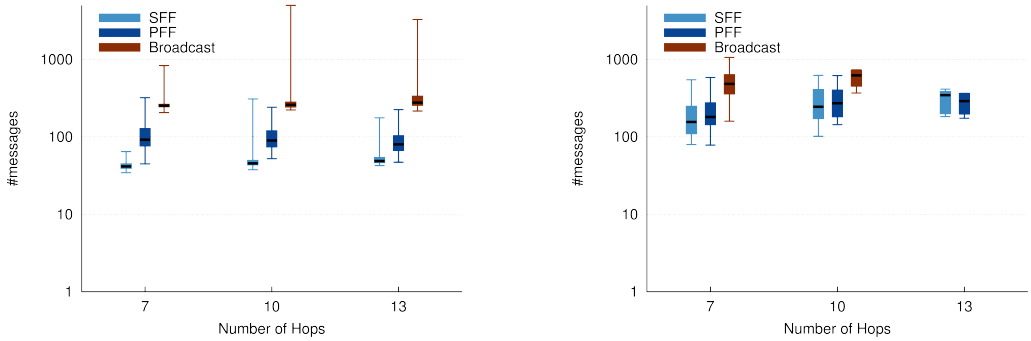
Figure 4.29: Circle Mobility Scenario - Content retrieval time

For the lower node density (Figure 4.29b), multi-hop forwarding is less efficient because there are many disruptions on the path. However, even in low density networks, Dynamic Unicast results in faster transmissions. For 7 hops, `SFF` strategy results in 3.63 times faster transmission than broadcast and for 10 hops it is 2.12 times faster than broadcast. For 13 hops, broadcast communication is not successful anymore but for Dynamic Unicast, some transmissions can still be completed. Thus, even in a low density environment, Dynamic Unicast leads to significantly faster content retrieval times because short end-to-end connectivity can better be exploited than with broadcast.

4.6.4 Number of Messages

In this section, we evaluate the message overhead of Dynamic Unicast compared to broadcast. The previous scenarios have shown that Dynamic Unicast results in faster throughput and fewer transmitted messages (no flooding). Figure 4.30 illustrates the transmitted Interests by mobile nodes. The x-axis denotes the number of hops, the y-axis the sent messages on an logarithmic scale. We can see that Dynamic Unicast results in fewer Interest messages than broadcast in all scenarios.

Figure 4.30a shows the high density network. We can see that SFF results in the fewest Interests, followed by PFF while broadcast sends considerably more Interests. For SFF, the number of transmitted Interests does not increase significantly for a higher path length, i.e., only 14.29% from 7 to 13 hops. For PFF, the number of transmitted Interests decreases with the path length by 13.04%. This may seem surprising, but the number of different paths does not increase for a higher path length in this scenario, however, the number of node does (to keep the same density). Thus, although the requester may send slightly more Interests, due to more nodes, the average is lower. With broadcast the number of transmitted Interests increases slightly by 9.5% from 7 to 13 hops. However, broadcast results in significantly more transmitted Interests than SFF even for 13 hops, e.g., 465% more Interests than SFF.



(a) 31.42m node density

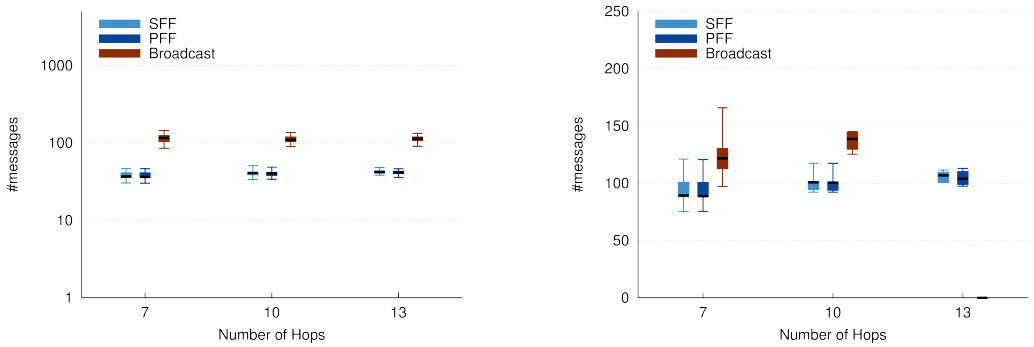
(b) 78.54m node density

Figure 4.30: Circle Mobility Scenario - sent Interests messages by mobile nodes.

For the lower node density (Figure 4.30b), Dynamic Unicast still sends fewer Interest messages than with broadcast, however, the difference is not so large anymore. Due to the lower node density, there are more broken paths in Dynamic Unicast, which leads to broadcast fallback. This increases the number of transmitted Interest messages.

Figure 4.31 illustrates the transmitted Data messages by the mobile nodes. In this graph the y-axis is not logarithmic. We can see that SFF and PFF perform similar and are both significantly better than broadcast. Please note that the hop distance has no impact on the average number of Data messages per node because the number of nodes has been adapted to keep the same density. Since all nodes follow a restricted mobility model in one direction, the

communication paths are more stable than in the previous scenarios.

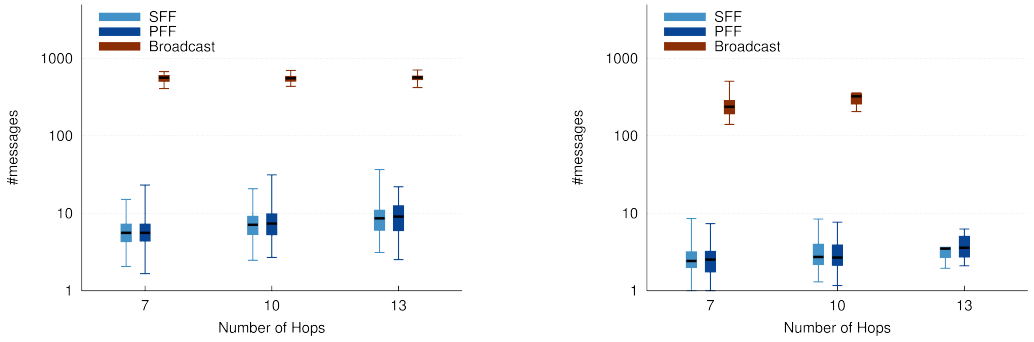


(a) 31.42m node density **(b)** 78.54m node density

Figure 4.31: Circle Mobility Scenario - sent Data messages by mobile nodes.

In a low density network (Figure 4.31b), `SFF` and `PFF` results in fewer Data transmission than with broadcast, however, the mobile nodes send slightly more Data messages than in high density networks (Figure 4.31a). Due to the lower node density, the total load is distributed over fewer mobile nodes, i.e., in high density networks fewer nodes perform Data transmissions because some nodes are not on the (unicast) path to the content source. Whereas having broadcast the number of Data messages stays approximately the same for high as well as low node densities.

Figure 4.32a shows the number of received duplicates at mobile nodes. We can see that Dynamic Unicast results in significantly fewer duplicates than broadcast in all scenarios. While `SFF` and `PFF` remain below 10 duplicates on average, broadcast requires 94 times more duplicates than `SFF` in the high node density scenario and 97 times more duplicates than `SFF` in the low node density scenario.



(a) 31.42m node density **(b)** 78.54m node density

Figure 4.32: Circle Mobility Scenario - Received duplicate Data messages by mobile nodes.

4.6.5 Summary

Dynamic Unicast performs also much better than broadcast in this scenario. If the node density is low, reliability of Dynamic Unicast is better than broadcasts because short periods of connectivity can be better exploited. `SFF` is up to 14.61 times faster than broadcast while having a lower message overhead.

4.7 Static Circle and Content-Requester-Tracker

If certain content is highly requested it can be advantageous for the source to send it via broadcast instead of multiple unicast connections. By doing so, it may be possible to send fewer messages and increase the overall throughput. Many simultaneous requests in the same content may result in a busy medium and reduce the unicast transmission speed for all requesters. Furthermore, a content source has to send the content multiple times to each requester resulting in more transmitted messages and a higher energy consumption. The idea of this evaluation is to find a threshold `R` of concurrent requests, where broadcast communication is more efficient. For further details of `CRT table` please consider section 3.7.

4.7.1 Topology and Parameters

The requester nodes are placed on a ring around the content source, such that each requester has the same distance to the content source. Figure 4.33 illustrates the topology.

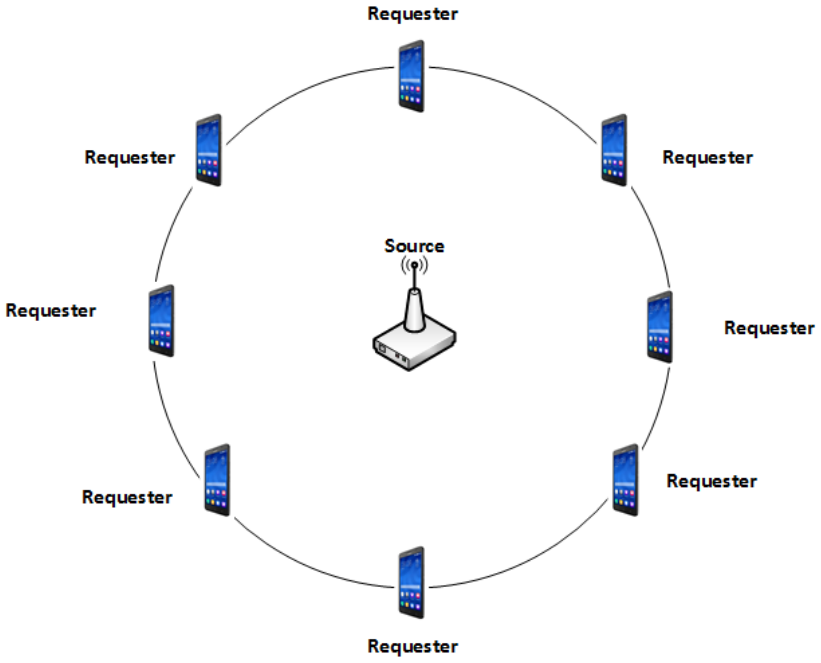


Figure 4.33: Sing-hop Circle Scenario - Topology

Four communication types were tested:

- **Unicast**
  Each requester has a single-hop unicast connection to the source.
- **Broadcast**
  Each requester requests the content via broadcast.

- **CRT Unicast**
  Each requester requests over unicast, and the source switches after `R` simultaneous unicast requests to broadcast.

- **CRT Broadcast Switch**
  Similarly to `CRT Unicast`, the source switches after `R` simultaneous unicast requests to broadcast. Each requester requests over unicast but switches to broadcast if the requested Data is received on the broadcast face.

Table 4.16 lists the used parameters for the `CRT` scenario. The `Topology` Section lists the topology specific settings. For this scenario a circle radius of 75m was chosen. Therefore, the distance between each requester an the content source is 75m. The wireless transmission range of the content source and the requesters was approximately 130m according the parameters in Section 4.2.2. Therefore, content source and requester are in direct connection. The number of requesters that retrieve the same file simultaneously is varied between 1 and 100. The `Simulation` section lists the simulation length (21600s). For this evaluation we use a static configuration, i.e., no mobility. The `CCNx` section list parameters such as file size (5MB) and the expire time. `Requester Max CRT` represents requesters threshold `N` to switch from unicast to broadcast requesting after `N` successively incoming broadcast Data messages. `Source Max CRT` describes sources threshold `R` to switch from unicast to broadcast.

Parameter	**Description**
Topology	
Circle Radius	75m
Number of Hops	1
Source Nodes	1
Number of Requests	1, 20, 50, 100
Simulation	
Mobility Model	No Mobility
Simulation Length	21600s (6h)
CCNx	
File Size	5MB (1281 content objects)
Communication Types	Broadcast, Unicast
Expire Time	180m (3h)
Requester Max CRT	2
Source Max CRT	40

Table 4.16: Content Requester Tracker (CRT) - Parameters

4.7.2 Content retrieval with disabled CRT

In this subsection we evaluate a suitable `R` for the `CRT table`. First, it has to be defined what the criterion for switching is. For example, a criterion could be the number of transmitted Data packets. But then, only broadcast communication would be used if more than 1 requester is asking for the content. Therefore, we select the download time as threshold criterion.

Content Retrieval Time

Figure 4.34 illustrates the content retrieval times for different simultaneous requests for a 5MB file. The y-axis denotes the time, the x-axis the number of simultaneous requests. Up to 20 simultaneous requests, unicast is advantageous compared to broadcast. For 50 simultaneous requests or more, broadcast content transmissions can be performed faster. We can observe that the broadcast transmission time stays constant independent of the number of requesters (1 or 100). However, transmission times for unicast increase from 1 requester (5.81s) to 100 requesters (221.76s) by 3717%. For 100 requesters, broadcast communication is 2.56 times faster than 100 simultaneous unicast requests. Based on this evaluation, we set the threshold `R` at a content source to 40 concurrent connections.

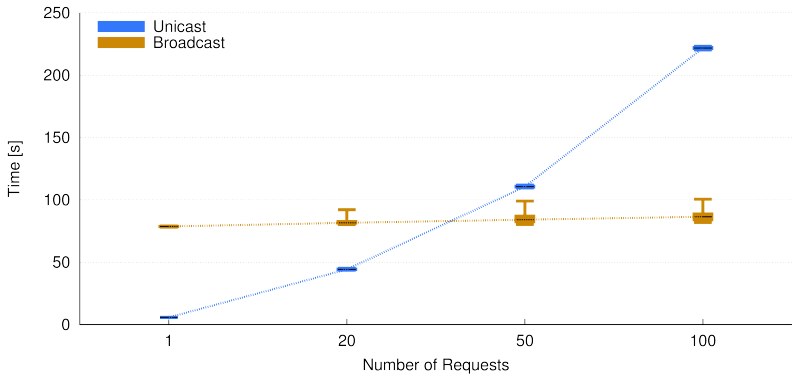


Figure 4.34: Content Retrieval Time, disabled `CRT`

Number of Messages

Figure 4.35 shows the received number of Interest messages at the source node. We can observe that the number of Interest messages (broadcast) at the content source is constant for an increasing number of requesters. If Interest messages are broadcasted, other nodes can overhear it and refrain from sending their own Interests. Consequently, if Interests are broadcasted, a content source receives only 15% more Interests with 100 concurrent requesters compared to a single requester. With unicast, however, nodes cannot overhear their transmissions, such that the Interests increase linearly with the number of requesters.

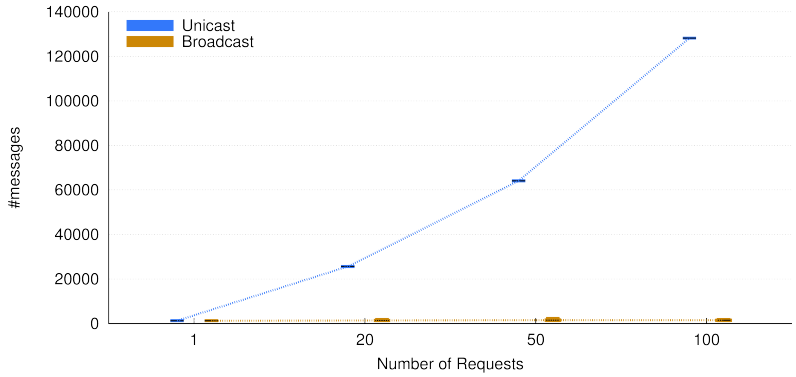


Figure 4.35: Received Interest messages by source, disabled `CRT`

This has also effect on the transmitted Data messages by the content source (Figure 4.36) since a content source replies to Interests with content. Consequently, if more Interests are received, more Data messages are replied. While for 100 concurrent requesters with broadcast only 2.97% more Data is transmitted compared to a single requester, 100 times more Data messages have been transmitted with unicast connections. As a consequence, throughput with 100 unicast requesters is lower than with broadcast.

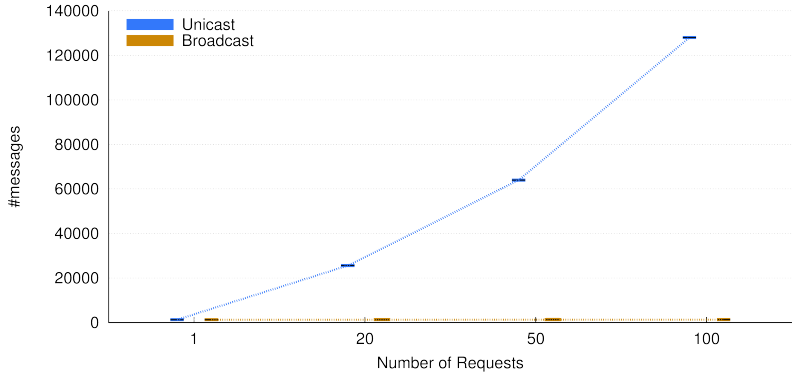


Figure 4.36: Transmitted Data messages by source, disabled CRT

4.7.3 Content retrieval with enabled CRT

In this section, we evaluate the `CRT` table for the threshold `R` (=40 concurrent requests) selected in the previous subsection. Four different communication types are compared:

- **Unicast** - Standard unicast as in the scenario before.
- **Broadcast** - Standard broadcast implementation.
- **CRT Unicast** - Content Source switches after 40 simultaneous requests to broadcast, whereas requester continues to request over unicast.
- **CRT broadcast switch** - Same as `CRT Unicast` with the addition that requesters switching back to broadcast when threshold N (=2) of subsequent incoming broadcast messages (in response to unicast requests) have been received.

Content Retrieval Time

Figure 4.37 illustrates the content retrieval times for the specified communication types. Until the threshold, `CRT unicast` and `CRT broadcast switch` perform exactly the same as unicast. If we pass the threshold, `CRT unicast` becomes worse than broadcast and `CRT broadcast switch`. This illustrates that it is not enough to switch communication to broadcast for specific number of requesters because requesters still send Interests via unicast. Thus, the increased number of Interests may result in collisions but Data is transmitted via delayed broadcast. Furthermore, we can see that `CRT broadcast switch` performs nearly equal to broadcast, i.e., only 22% more time than broadcast for 100 concurrent requests, because requesters switch to broadcast transmission as well. Compared to unicast, `CRT broadcast switch` performs 2.10 times faster (having 100 concurrent requests).

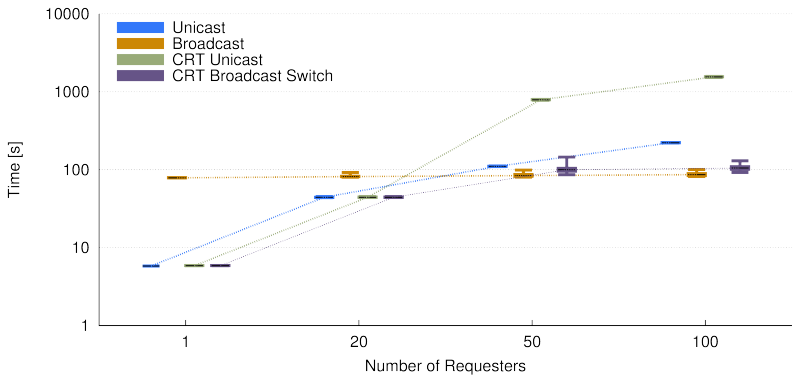


Figure 4.37: Content Retrieval Time, enabled `CRT`

Number of Messages

Figure 4.38 illustrates the number of received Interest messages at the source node. `CRT unicast` sends as many Interests as unicast. This is due to the missing broadcast switch

on requester nodes. Considering `CRT Broadcast switch` the number of received Interests is equivalent to unicast below the threshold `R` and slightly higher than broadcast above the threshold.

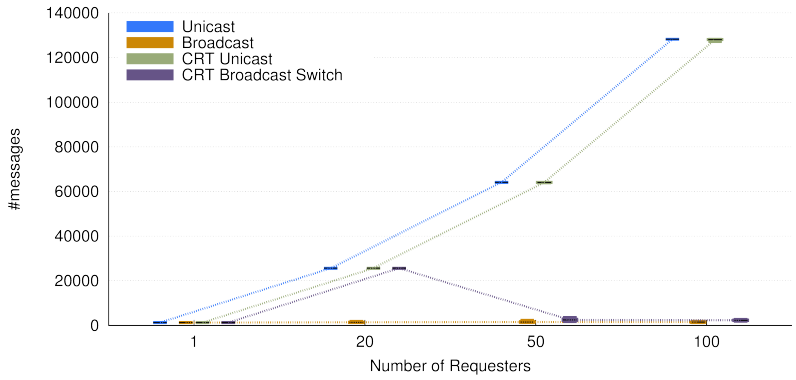


Figure 4.38: Received Interest messages by source, enabled `CRT`

Figure 4.39 shows the number of transmitted Data messages by the source. For 100 concurrent requesters, `CRT unicast` sends 21.46 times more Data messages than `CRT broadcast switch`. The content source has no memory of what content has already been transmitted but every unicast request is answered via broadcast. Thus, if the content source receives significantly more Interests, more Data transmissions are performed. As a result, the content source sends 25000 duplicates with `CRT unicast`, while only 133 duplicates are send with `CRT broadcast switch` although unicast Interest messages are answered via broadcast.

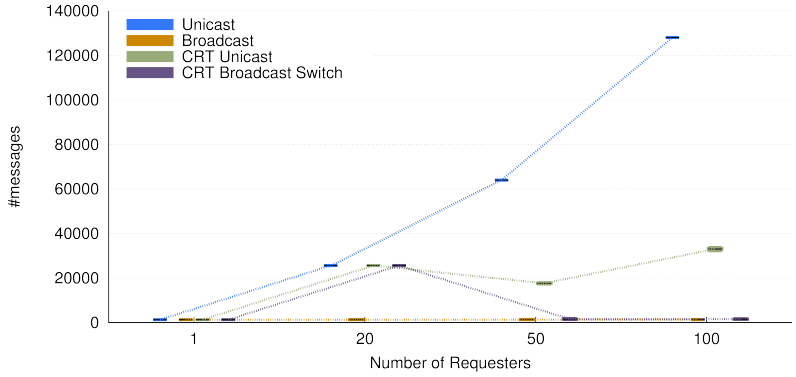


Figure 4.39: Transmitted Data messages by source, with enabled `CRT`.

4.7.4 Summary

With enabled `CRT` and 100 simultaneous requests, throughput of a 5MB file can be increased in best case by 110.92% compared to standard unicast. Meanwhile, at content source, the overhead of Data message can be reduced up to 99.47%.

Defining a threshold `R` to switch from multiple unicast connections to broadcast is complex. It depends on the selected decision criterion, e.g., transmission time or transmitted messages.

Furthermore, it is not enough to switch only the content source to broadcast for multiple unicast requests because the requesters still transmit unicast Interest. Thus, requesters also need to switch to broadcast requests when they receive broadcast answers to unicast requests. However, if requesters switch to broadcast, the content source has no means to decide whether all requesters are still interested in the content (where broadcast would make sense) or lost their interest in it. Particularly, in mobile environments where connectivity changes frequently, the decision (broadcast or unicast) cannot be done once but need to be rechecked periodically.

Furthermore, broadcast requests may also be beneficial if nodes do not request the content exactly at the same time but slightly time shifted (the content may already be in their caches). Thus, heuristics for content popularity may help in this situation.

Overall, we can conclude that switching from unicast to broadcast is not an easy mechanism. If not done properly, it results in worse performance.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Broadcast adaption uses broadcast to find a content source and establish a route to the content source when Data packets travel the reverse path back to the requester. Evaluations have shown that this strategy is also very efficient in mobile scenarios but it is required to delete expired information quickly. In every tested scenario broadcast adaption showed better performance than broadcast. However, if the node density decreases (or for long paths with high mobility), unicast paths break more often and broadcast adaption performs similar to broadcast (due to more frequent fallbacks to broadcast).

In a scenario with `static source and requester and mobile forwarder nodes`, broadcast adaption performs better than broadcast. The `SFF` strategy is up to 46.66 times faster than broadcast (Pedestrian Scenario with single request). Because broadcast requires additional forwarding delays for duplicate suppression and has lower data rates than unicast, broadcast adaption can result in faster transmissions in high distance and low density communication. Furthermore, broadcast adaption results in much lower message overhead. Depending on the scenario, mobile nodes that use `SFF` send up 12 times less Data messages and up to 6.06 time less Interests than with broadcast because messages are not flooded in the network. Surprisingly, broadcast adaption performed even better for highly requested content up to 32 requesters. Although broadcast adaption may send a few Interests more, fewer data messages are sent compared to broadcast because content can be found in the nearest cache and does not need to be retransmitted over the entire path. Evaluations have shown that `SFF` is mostly more efficient than `PFF` because fewer Interests are transmitted and, consequently, the wireless medium can better be utilized for Data transmissions. Only in high distance low density topologies, it can happen, that `PFF` performs slightly better because of a higher path redundancy.

In a `grid scenario` with low content source density, i.e., no coverage over the entire playground, multi-hop communication performs much better than single-hop because requesters do not need to go in the content sources' wireless transmission range and can request content from far away. Only if the entire playground is covered with content sources, single-hop communication is more efficient due to fewer network traffic. Multi-hop communication with

broadcast adaption is very efficient in terms of content retrieval time and message overhead in a grid scenario. If communication is broadcast only, forwarders send on average 5 -10 times more Interest messages than with broadcast adaption and 2-7.5 time more data messages. Furthermore, because Interest messages are directed towards a specific content source (and not all content sources as with broadcast), content sources send 68.82% fewer Data messages with broadcast adaption compared to broadcast (30 concurrent requests, 16 sources). Consequently, the number of duplicates received by mobile nodes is 97.41% lower with broadcast adaption compared to broadcast (30 concurrent requests, 16 sources).

Furthermore, broadcast adaption performed also better than broadcast in a `circular mobility scenario` with pedestrian mobility speed. Particularly, if node density is high, broadcast adaption is better than broadcast because short periods of connectivity can be better exploited. In this case, `SFF` is up to 14.61 times faster than broadcast while having a lower message overhead. However, for high (vehicular) node mobility broadcast adaption is only better up to a certain path length. For longer path lengths, broadcast adaption results in frequent path breaks such that broadcast performs slightly better.

In scenarios with many `concurrent unicast requesters`, broadcast transmission may be considered more efficient. In such a scenario, a `CRT table` may lead to performance gain (transmission times and number of transmitted messages). However, in mobile networks the application of `CRT` is not easy and depends on many factors. For example, both requester and content sources need to switch from unicast to broadcast transmission. It is not enough to just let the content source respond to unicast requests with broadcast. Thus, if not done properly, switching from unicast to broadcast may result in worse performance.

Overall we can conclude, that broadcast adaption can decrease content retrieval time and message overhead compared to broadcast in mobile environments.

5.2 Future Work

5.2.1 Adaptive CCNx Parameters

In the current evaluations, the default Interest lifetime of 4 seconds was used. The Interest lifetime has a direct impact when path breaks (timeouts) can be detected. Therefore, in mobile networks where paths break frequently, this may be a too high value. It should be possible to set the Interest lifetime dynamically, e.g. based on the round-trip time. This may result in significantly faster content transmission times in highly mobile networks.

Furthermore, an adaptive mechanism to set the pipeline size depending on the mobility (e.g., a large pipeline size in static networks and a low pipeline size in mobile networks) could help to increase throughput.

5.2.2 Location Based Routing

The current routing strategies `SFF` and `PFF` randomly forward the Interests either over the best face or all available unicast faces. The path is selected based on an initial broadcast flooding. The nodes that respond with data are random and not optimized. Therefore, location based routing may help to select better (or even optimal) forwarders between requester and content source. Forwarder nodes could for example include their current position, speed and direction in broadcast messages transmitted in the network. Based on this information, a requester node could calculate the best neighbor node for routing.

5.2.3 CRT

Currently the `CRT`-table is just a proof of concept implementation. In potential future implementations for mobile networks the following issues need to be explored further:

- **Context based role** - distinguishing between requester and content source: Currently, we have two separate implementations for requesters and content sources, however, in practice a node can be both (for different content) at the same time. Thus, future implementations should include more flexibility to switch the context based on the role (content source or requester).
- **Security** - Since deletion of a `CRT` entry at a content source depends on the transmission of the last segment, a malicious requester could easily force a `CRT` entry deletion by requesting only the last segment (given he knows the segment number). Thus, different deletion/aging criterion may be evaluated.
- **Extension for mobile networks** - Currently, the `CRT table` has only been evaluated in a static environment where the number of requesters does not change. However, in a mobile network, connectivity and, thus, also the number of concurrent requesters may change frequently. Thus, decisions to switch from unicast to broadcast may need to be re-evaluated in dynamic environments.

Chapter 6

Appendix

6.1 Pedestrian node speed with 4 pipeline size

The results of this evaluation can be found on the accompanying DVD (/experiments/results/-graphs/3_Static-Mobility/1_single-requester/1.2ms/4-Pipeline/).

6.2 Pedestrian and vehicular node speed, keeping node density

Figure 6.2 shows the content retrieval time if node density is kept by $2500m^2$ per node compared with $5625m^2$ per node having pedestrian node speed. With higher node density content retrieval performs with `SFF` 6.87 times better than with low node density. Same effect using `PFF` or broadcast.

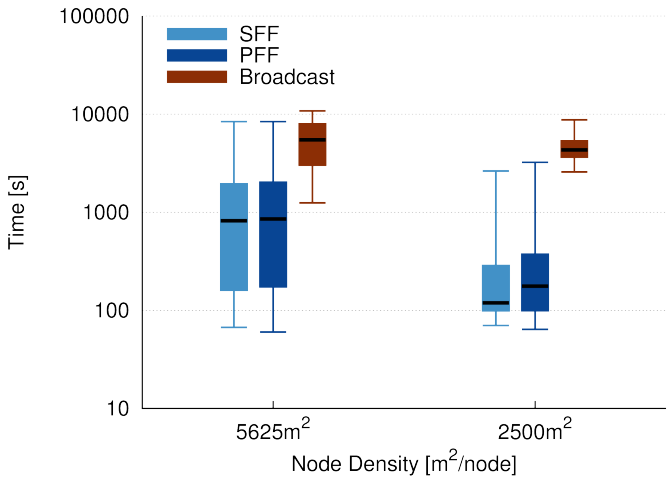


Figure 6.1: Multi-hop communication with static source and requester - pedestrian speed with distance 750m, low and high node density.

Figure 6.2 shows the content retrieval time if node density is kept by $2500m^2$ per node compared

with $5625m^2$ per node having vehicular node speed. With higher node density content retrieval performs with SFF 2.69 times better than with low node density. Same effect using PFF or broadcast.

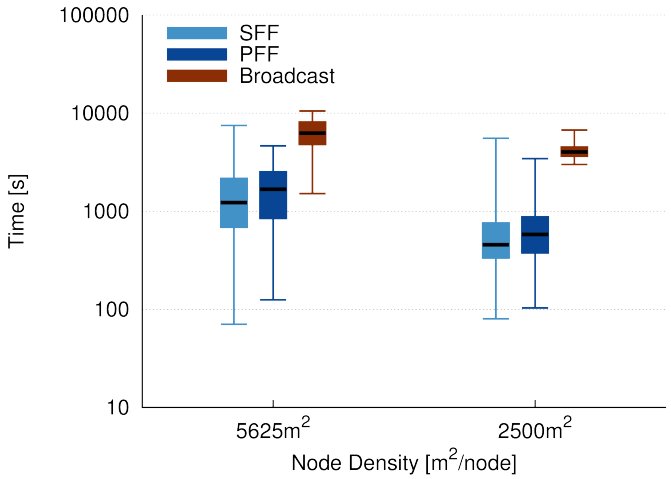


Figure 6.2: Multi-hop communication with static source and requester - vehicular speed with distance 750m, low and high node density.

6.3 Further Evaluation Results

All additional results can be found on the accompanying DVD. Please consider section `/experiments/results/`

6.4 Source Code

The source code of the implementation can be found on the accompanying DVD: `/src/CCNx/`

You will find 3 builds:

- **ccnx-0.8.2** - This is the original CCNx version. Based on that the implementation of this thesis were done.
- **ccnx-0.8.2-CRT** - This is the implementation containing the CRT extension.
- **ccnx-0.8.2-Dynamic-Unicast** - This extension contains the Dynamic Unicast functionality. Furthermore, it contains the resume capabilities, which was implemented on the scope of another work [11].

Furthermore you will find on DVD manuals about how to install and build the CCNx versions.

Bibliography

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[2] J. Weber, "Automatic detection of forwarding opportunities in intermittently connected content-centric networks," 2013.

[3] *NS3-DCE (Direct Code Execution)*. [Online]. Available: https://www.nsnam.org/overview/projects/direct-code-execution/

[4] *Project CCNx*. [Online]. Available: http://www.ccnx.org/

[5] *CCNx Daemon*. [Online]. Available: http://www.ccnx.org/releases/latest/doc/manpages/ccnd.1.html

[6] *Routing daemon ccndc*. [Online]. Available: http://www.ccnx.org/releases/latest/doc/manpages/ccndc.1.html

[7] *CCNx Face Management and Registration Protocol*. [Online]. Available: https://www.ccnx.org/releases/latest/doc/technical/Registration.html

[8] *CCNx Strategies*. [Online]. Available: https://www.ccnx.org/releases/latest/doc/technical/CCNxProtocol.html

[9] "Alix3d2 system board." [Online]. Available: http://www.pcengines.ch/alix3d2.htm

[10] *Requester Application ccncat*. [Online]. Available: https://www.ccnx.org/releases/latest/doc/manpages/ccncat.1.html

[11] T. Schmid, "Agent-based data retrieval for oportunistic content-centric networks," Master's thesis, 2015.

[12] *NS3 Random Waypoint Mobility Model*. [Online]. Available: https://www.nsnam.org/docs/release/3.10/doxygen/classns3_1_1_random_waypoint_mobility_model.html

[13] *NS3*. [Online]. Available: https://www.nsnam.org/

[14] *UBELIX - University of Bern Linux Cluster*. [Online]. Available: http://www.id.unibe.ch/content/services/ubelix/overview/index_ger.html

[15] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," RFC 3376 (Proposed Standard), Internet Engineering Task Force, October 2002, updated by RFC 4604. [Online]. Available: http://www.ietf.org/rfc/rfc3376.txt