Analysis and Improvement of Name-based Packet Forwarding over Flat ID Network Architectures

Antonio Rodrigues
adamiaonr@cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA
Faculty of Engineering, Univ. of Porto
Instituto de Telecomunicações
Porto, Portugal

Peter Steenkiste
prs@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Ana Aguiar
anaa@fe.up.pt
Faculty of Engineering, Univ. of Porto
Instituto de Telecomunicações
Porto, Portugal

ABSTRACT

One of ICN's main challenges is attaining forwarding table scalability when confronted with a huge content space. This is specially true in flat ID architectures, which do not naturally support route aggregation for hierarchical names. Previous work has proposed improving scalability by aggregating content names in a fixed-size Bloom Filters (BF). However, the negative impact of false positive (FPs) matches on forwarding correctness and performance has not been studied thoroughly. In this paper, we study the end-to-end network performance of BF-based forwarding. We devise an analytical model that accurately models BF-based forwarding and the flow of BF-based packets over inter-domain topologies. Using this model, we show that the use of BFs for packet forwarding puts feasibility at odds with scalability. Based on our analysis, we propose and evaluate several strategies to improve the feasibility of BF-based forwarding and examine different scenarios for which BF-based forwarding is suitable.

This paper is an extended version of our conference publication [17]. For citation purposes, please cite the conference version.

KEYWORDS

ICN; Bloom filter; forwarding, false positive

1 INTRODUCTION

Hierarchical names are widely used by network applications to identify content and services: e.g., data retrieval in the Web uses URLs to identify web objects. Name-based Information Centric Networks (ICNs) such as NDN [7] promise to be a good fit for such applications, for two reasons: (1) no need for external name lookup (e.g., DNS or GNS [18]) by using name-based forwarding at the network layer; and (2) reduced forwarding table sizes as a result of the aggregation enabled by the hierarchical structure of names. In contrast, ICNs based on flat IDs (defined as fixed-size bit strings in this paper), e.g., DONA [11], PURSUIT [5], do not natively have these benefits.

Recently, a forwarding scheme based on Bloom Filters (BFs) [12, 13] has been proposed to enable aggregation of content descriptors (e.g., tag sets, hierarchical names, etc.), thus improve the scalability of flat-ID ICNs. We henceforth refer to this scheme as 'Bloom Filter-based forwarding'. This method also enables NDN-style forwarding on flat ID-based ICNs, in the sense that it allows for the encoding of hierarchical names of variable size in a fixed-sized bit array: the BF itself. Notwithstanding the routing scalability benefits of this method, false positive (FP) matches - an unavoidable side-effect of BFs - can severely impact *forwarding correctness*.

In this paper we study the impact of FP matches on the correctness of BF-based forwarding. We show that the exclusive reliance on BFs for forwarding in a distributed network raises a trade-off between scalability (attained via prefix aggregation) and forwarding correctness (affected by FP matches), which has not been studied in previous work. This trade-off depends on: (1) the degree of prefix aggregation in forwarding tables; (2) the number of entries in forwarding tables; and (3) the BF size (in bits). In particular, we show that high degrees of aggregation result in an higher likelihood of forwarding errors, particularly when BFs used in forwarding tables are created from short names (e.g., '/acme.org') and BFs used in packets are created from long names (e.g., URLs with 10 components, a reasonable number to expect in practice [12, 14]).

This paper makes the following contributions:

- We define a model of BF-based packet flow, which can be used to quantify the effect of FP matches on BF-based forwarding, over arbitrary network topologies. The model uses a Bayesian Network approach to calculate the probability of forwarding decisions at a BF-based router, thus capturing the network-level effects of FP matches.
- We evaluate the impact of FP matches on BF-based forwarding, with focus on the trade-off between scalability and forwarding correctness. The evaluations uses PoP-level Rocketfuel topologies [20]. Our results show that when aggressive aggregation is used, BF-based forwarding incurs excessive incorrect deliveries and link usage, when compared to forwarding with exact match semantics. Larger BFs must be used to get acceptable performance.
- We propose extensions to 'vanilla' BF-based forwarding, which aim at reducing the negative impact of FPs: *XIA-style fallbacks* [6] and *prefix-exclusion*. The former allow routers to quickly switch to locator-based forwarding if BF-based forwarding fails, reducing latency costs of wrong deliveries; the latter exploits the design of namespaces to reduce FP rates while maintaining relatively short BF sizes.

The remainder of the paper is organized as follows. In §2 and §3, we describe BF-based forwarding analyze the impact of FP matches on forwarding in a single router. We present an FP-aware forwarding design in §4. We describe our BF-based packet flow model in §5, using it to evaluate BF-based forwarding in §6. Finally, we discuss related work and conclude.

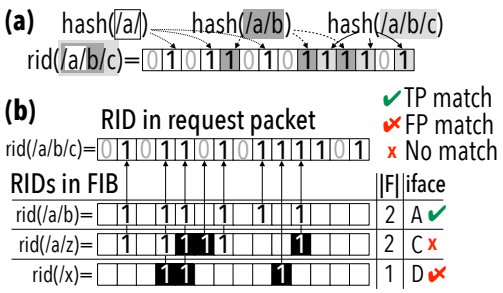

Figure 1: (a) Name encoding; (b) RID lookups between a request and multiple forwarding entries.

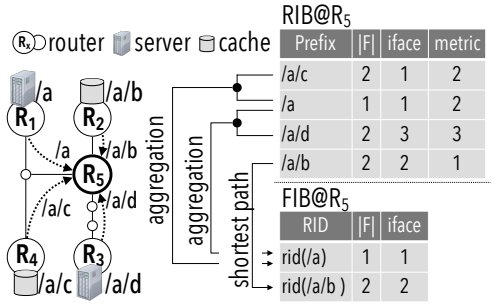

Figure 2: Example of RID routing and prefix aggregation rules.

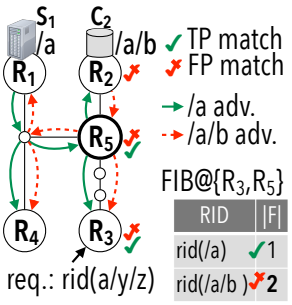

Figure 3: Examples of multiple matches and wrong tree bindings.

2 FORWARDING WITH BLOOM FILTERS

In this section, we provide the background on the Bloom Filter (BF) based forwarding scheme studied in this paper.

2.1 Name Encoding and BF-based Lookup

Name encoding: Our encoding method is a specialization of that used in TagNet [12, 13]), which uses Bloom filters (BFs) [1] to represent sets of string tags at the network layer. While the tags encoded in TagNet are independent (e.g., keywords), we instead encode all the *sub-prefixes* of a hierarchical name. We refer to the resulting identifiers as **Request IDs (RIDs)**, while we call the number of sub-prefixes the length or size of the name. Fig. 1(a) shows how the hierarchical name '/a/b/c' is encoded based on its three sub-prefixes {'/a', '/a/b', '/a/b/c'}. This is done using a function `rid(name)` which partitions `name` into its sub-prefixes, each of which is encoded in m bit BF using k hash functions, forming the RID.

This form of encoding offers many benefits. First, network nodes that hold all the content in a namespace subtree can generate a single aggregate announcement, thus reducing forwarding table sizes. Second, applications can generate RIDs locally based on a name, without the help of a lookup system, unlike other schemes [5, 9, 18]. Third, BFs allow for the encoding of names of variable size into a fixed size bit array, which is needed for flat ID network (e.g. DONA [11], MobilityFirst [18] or XIA [6]). However, there is a limit to the name size, since longer names result in higher false positive (FP) rates (see §3). Finally, encoding sub-prefixes instead of name elements (i.e., the portions in-between '/') reduces FP rates, since that enforces the correct ordering of name components, e.g., '/a/b/' and '/b/a/' map to different RIDs.

Lookups: RIDs are used as the intent address in content requests packets, similar to names in NDN Interest packets [7]. Entries in forwarding tables also include an RID. As shown in Fig. 1(b), forwarding table entries contain three fields: (1) an RID F, created from a name such as '/a/b'; (2) the size of the name encoded in the RID, $|F|$ (2 in the case of '/a/b'); and (3) an interface. A lookup between R (the RID in the request packet) and F (the RID in a table entry) consists in checking if the set of sub-prefixes encoded in F is a subset of those encoded in R. Similarly to TagNet [12, 13], a lookup can be implemented as a bitwise inclusive AND (&) in-between R and F, i.e., $(F \,\&\, R) == F$. As shown in Fig. 1(b), the lookup can result in false positive (FP) matches. The black bits in the FIB entries are set by hashing sub-prefixes not present in request R. This results in a (correct) "no match" for the second entry, but a FP for the third entry. We analyze the effects of FP matches in §3.

2.2 Routing & Prefix Aggregation

Content can be replicated throughout the network: e.g., origin servers, CDN nodes, and caches which opportunistically cache content. Such nodes announce names of different sizes. Origin servers announce short prefixes (e.g., '/a'), since they have a copy of all the data. CDNs may store of a subset of the data, in which case they announce longer names (e.g., '/a/b'). Finally opportunistic caches store individual objects and thus announce full names. Routers use a routing protocol to identify the "closest" source for each name, and the announcements they receive with different prefix sizes and costs, not unlike how CIDR addresses are advertized by BGP. Similar to IP, we use a forwarding engines with Longest Prefix Matching (LPM) semantics based on forwarding tables generated by the routing protocol.

Example: Fig. 2 shows an example of how routing and prefix aggregation can be handled in a BF-based forwarding system. Routers keep a Routing Information Base (RIB) and a Forwarding Information Base (FIB): RIB entries are prefixes in their textual form, as received from route announcements, while FIB entries hold RIDs, as explained earlier. Say a router R receives a pair of prefix announcements, each with different associated costs, for example path lengths, measured in hops. R will insert FIB entries for both prefixes if the cost of the longer prefix is smaller than that of the shorter prefix. E.g., the case of the pair {'/a', '/a/b'} in Fig. 2. Otherwise, only the shorter prefix is added, e.g., the case of pairs {'/a', '/a/c'} and {'/a', '/a/d'}.

Spanning trees: Paths created by shortest path routing form a spanning tree rooted at the content source. E.g., the tree for prefix '/a' advertised by S_2, in Fig. 3. Shortest-path spanning trees are implicitly created by the forwarding entries in the FIBs, which differs from TagNet, which explicitly uses trees for forwarding [12, 13]. Nevertheless, our subsequent analysis also applies to schemes such as TagNet, within the context of a single spanning tree.

Aggregation gains: As illustrated in Fig. 2, the RID design extends the aggregation gains provided by hierarchical names to flat ID architectures. Further gains could be attained by aggregating independent prefixes associated with the same egress interface into a

Matched	FP vs. TP match size		
interfaces	$\|FP\| < \|TP\|$	$\|FP\| = \|TP\|$	$\|FP\| > \|TP\|$
Different	Correct	FP detected	Incorrect
Same	FP detected	FP detected	FP detected

Table 1: Forwarding correctness outcomes.

single BF, e.g., {'/a'} $\rightarrow i$ and {'/b'} $\rightarrow i$ could be aggregated into a single BF as {'/a', '/b'} $\rightarrow i$. Similarly to TagNet [12, 13], we do not consider this type of aggregation since it further increases FP rates of the basic design (Fig. 1), which already have significant impact as discussed in §3.

3 WHY DO FALSE POSITIVES MATTER?

We look at the network-level impact of false positive (FP) matches and quantify router-level FPs as a function of aggregation levels.

3.1 Network-level Impact of False Positives

We look at the impact of FPs both at the level of routers and networks. We assume that forwarding tables are *complete*, i.e., there is a guarantee of at least one True Positive (TP) match for a given request at any forwarding table.

Multiple Matches: In contrast to exact-match forwarding, lookups in BF-based forwarding result in a mix of correct (TP) and incorrect (FP) matches. Under Longest Prefix Matching (LPM) semantics, the correctness of a BF-based forwarding decision can be characterized by (1) the length of the longest TP and FP matches; and (2) the egress interfaces associated with the longest matches. The different outcomes are summarized in Table 1. If two (or more) matches have the same length or are associated with the same interface, a FP match can be detected, since the routing protocol would never create such TP entries. When no FP match is detected, routers cannot know whether they are forwarding the packet correctly, because of LPM semantics. E.g., if two matches are associated with different interfaces, the packet is incorrectly forwarded when the FP match is longer than the TP match. If the FP match is shorter than the TP match, the packet is correctly forwarded.

Other options exist, such as multiple TP matches of different lengths and with different egress interfaces. These result in the same three types of outcomes listed in Table 1. Routers can use a number of strategies when they detect an FP, as we discuss in §4. Note that if the FP and TP refer to the same egress interface, the obvious strategy is to simply forward the packet through that interface. However, that is unlikely to happen in a later router, as we discuss next.

Wrong tree bindings: When looking at the impact of FPs on the correct delivery of packets, one must consider that forwarding tables in routers are not independent. Instead, they represent an (implicit) set of spanning trees rooted at content sources, as discussed in §2. Matches (either TP or FP) result in packets being forwarded along such trees towards the root. As a result, an FP match in one router can bind an RID packet to a spanning tree. The packet is then forwarded towards the wrong content. This is worrisome, since it means that a FP match on any router along the correct path can deviate the packet towards an incorrect delivery.

In the example in Fig. 3, two content sources (S_1 and C_2) announce prefixes along the green and red spanning trees. An RID request for '/a/y/z' entering router R_3 is matched both by a TP entry of size 1 (encoding '/a', announced by S_1) and a FP entry of size 2 (encoding '/a/b', announced by C_2), in both R_3 and R_5. Due to LPM rules, the request packet will be incorrectly forwarded by R_5 towards C_2, since the longer prefix is advertised along the spanning tree rooted at C_2. Unfortunately, C_2 is the wrong content source.

Incorrect deliveries will generally delay content retrieval. For example, the node receiving the incorrectly delivered request is likely to drop it, and the requester must rely on a timeout to recover. The requester must then reissue the request using a different type of address, since using the same RID is likely to result in the same incorrect outcome. We consider different recovery methods in §4. Given the high penalty of incorrect deliveries, it is important to keep router-level FP rates low.

3.2 False Positive Rates

As a first step towards evaluating the impact of forwarding errors, we quantify FP rates at two scopes - single forwarding entry lookup (§3.2.1) and full table lookup (§3.2.2).

3.2.1 False Positive Rate for a Single Lookup. The properties of Bloom filters (BFs) have been studied extensively [1]. The false positive (FP) rate for a BF of size m bit, encoding n elements with k hash functions is given by

$$fpr = (1 - (1 - 1/m)^{kn})^k \approx (1 - e^{-(kn)/m})^k \quad (1)$$

The value of k which minimizes the FP rate for given a $\{m, n\}$ pair is

$$k = \ln(2)\, m/n \quad (2)$$

The FP rate for a single RID lookup - i.e., verifying whether a request R matches a single forwarding entry F - can be derived from Eqs. 1 and 2 and expressed as Eqs. 3 and 4:

$$p(|R|, |F \backslash R|) = \left[\left(1 - e^{-(k|R|)/m}\right)^k\right]^{|F \backslash R|} \quad (3)$$

$$k = \ln(2)\, m/|R|_{max} \quad (4)$$

Similar expressions are provided in [1, 12]. The key insights to take away from Eqs. 3 and 4 are:

FPs can only be caused by prefixes that are different in the entry and the request. Since we have $|R|$ inner prefixes encoded in the request's BF, we replace n in Eq. 1 with the request size, $|R|$. Prefixes that are common between a table entry F and the request R are effectively true positives (TPs), and thus do not cause FPs. However, the prefixes encoded in F but not in R can result in FPs. We define the number of non-common prefixes as the 'request-entry difference', and represent it as $|F \backslash R|$. E.g., the request '/rk/csr /spk' and entry '/rk/ppr' would yield $|F \backslash R| = 1$.

All network entities must use the same hash functions and BF size. This relates to Eq. 4 and $|R|_{max}$, the maximum number of name elements that can be encoded in a request. To ensure a correct comparison between requests and entries, all network entities - endpoints, routers - must use the same BF size m as well as the same type and number of hash functions k. Network entities

cannot pick these parameters independently, so we fix k in our system by assuming a fixed value for $|R|_{max}$.

Note that for a fixed BF size, the maximum limit $|R|_{max}$ directly influences the number of hash functions used in BF encoding and membership tests, and thus FP rates. The larger the $|R|_{max}$, the smaller the number of hashes, and thus the larger the single lookup FP rate. This points us towards a simple way of limiting FP rates: ***prefix exclusion***. Instead of encoding each and every inner prefix of a name in BFs, one can limit the number of prefixes we encode. More generally, we can discard certain prefix lengths from the namespace when generating RIDs. E.g., for the name '/a/b/c', we could generate RIDs using only 2 out of the possible 3 prefix lengths, say '/a' (length 1) and '/a/b/c' (length 3), excluding the prefix '/a /b' (length 2). However, excluding prefix lengths comes at the price of increased forwarding table sizes. In order to ensure forwarding correctness, a forwarding entry may have to be replaced by multiple entries to account for the missing prefix length. E.g., if one excludes prefix length 2, all prefixes of length 2 encoded in forwarding tables must be replaced by all the underlying prefixes of length 3. This may significantly increase forwarding table sizes, and in turn increase FP rates. As such, prefix exclusion should only be used when the associated FP rate reduction is not offset by the FP rate increase resulting from larger forwarding tables.

3.2.2 Router-level False Positive Rate. Ultimately, we are interested in the probability of having FP matches on a lookup on a forwarding table, e.g., composed by $\sim 10^7$ entries, as considered in recent ICN research [13, 15, 22]. Assuming that the occurrence of FPs is independent across entries, the probability of having *at least one* FP match in a table can be derived from Eq. 3 as in Eq. 5:

$$P = 1 - \left(\prod_{j=1}^{|R|} (1 - p(|R|, |F \backslash R| = j))^{\Phi(j)} \right) \qquad (5)$$

$\Phi(j)$ = # of fwd. entries with $|F \backslash R|$ = j

The term within the product of Eq. 5 represents the probability of having no FP matches among all entries with request-entry differences equal to j. As such, the product represents the probability of having no FP matches among *all entries* in a forwarding table. Eq. 5 is sensitive to large single-lookup FP rates, since that makes the term within the product approach 0.

Router-level FP rates vs. aggregation: In Fig. 4(a) we use Eq. 5 to show how the router-level FP rate varies with different aggregation levels. We consider a table with 10^7 entries, each encoding a variable numbers of inner prefixes: from 1 to 10. We consider 3 different table compositions: a table mostly composed by (1) short entries (63% of entries of size $|F| = 1$, 30% of size $|F| = 2$); (2) large entries (30% of size $|F| = 9$, 63% of size $|F| = 10$); and (3) a uniform distribution of entry sizes (10% for all size $|F|$). The BF size is set to 192 bit, as in [12, 13]. We consider that the lookup is made for a request size of 10. The occurrence of at least one FP match is guaranteed in almost all scenarios, except when most entries are long and the request-entry differences are large. Fig. 4(b) shows that for the 'short entry' scenario, the BF size must increase from 192 to 512 bit to reduce router-level FP rates below 10^{-1}.

3.3 Forwarding Errors, in Practice

Fig. 5 provides practical evidence of the aforementioned errors. We create a forwarding table with 2×10^6 entries, sizes 1 to 10. The entries are built from names picked from a sample of the `web09-bst` URL dataset [8], composed by ~ 20 million URLs. The distribution of forwarding entry sizes is given in Fig. 5(a). We lookup 5000 different BF requests of size 10 against the table. The name used in each request is built by picking one prefix encoded in the table, and adding the necessary number of suffixes (words picked from a dictionary, at random) to complete it till size 10. The lengths of prefixes used to generate requests are picked with uniform probability, 500 prefixes per each size (1 to 10). This way, each request is guaranteed to have a TP match, and all TP match sizes are tested. The size of BFs used to encode prefixes and requests is 192 bit, as in [12]. For each request, we record the number of FP matches larger than or equal to the largest TP match: e.g., if a request matches 2 (or more) TP entries, we only consider the largest TP match for comparison.

The distribution of FP match sizes does not follow that of the forwarding entries. Fig. 5(b) shows that more than 90% of FP matches occur for the shortest entry sizes, 1 or 2. This shows how FP matches are more prevalent among highly aggregated entries. In absolute terms, size 1 FPs account with $\sim$ 280k matches, size 2 with $\sim$ 15k, for $|FP| = |TP|$. For $|FP| > |TP|$, size 1 accounts with $\sim$ 16k matches, size 2 with 154. Larger FP match sizes are negligible. Furthermore, the most common problem is having FPs with the same size as TPs. Errors caused by FPs larger than TPs occur less frequently.

4 FP-AWARE FORWARDING ENGINE

Based on our understanding of forwarding errors caused by false positive (FP) matches (§3.1), we now describe a BF-based forwarding engine which follows Longest Prefix Matching (LPM) forwarding semantics, and is capable of partial detection and resolution of FP matches. We consider this BF-based forwarding engine for the remainder of this paper.

4.1 Overview

Fig. 6 describes the FP-aware BF-based forwarding algorithm we propose. It is similar to the algorithm given in §2, as it forwards request packets over the interfaces associated with the longest match. The difference lies in the partial detection of FP matches, namely by identifying the case of 'multiple matches' of the same length at different interfaces (see §3.1 and Table 1).

Algorithm flow: After a table lookup, (step 1), the algorithm collects the set of interfaces associated with the longest matches (step 2), the *interface set* S. The cardinality of the interface set can be $|S| \geq 1$, since we assume that forwarding tables are complete: as such, empty interface sets - i.e., $S = \{\emptyset\}$ - are not possible. If the set includes the local interface (step 3), or if it includes a single egress interface (step 5.2), the packet is simply delivered to the local cache (4), or forwarded (step 5.2). Otherwise, we execute a *'multiple match' resolution* (MMR) strategy (step 5.1). We discuss different MMR strategies in §4.2. In the case of a local cache delivery, the packet can either be delivered correctly (step 4.1) or incorrectly (4.2). Caches can differentiate the two cases if packets carry the full names, e.g., as part of the application-layer metadata. If an incorrectly delivery is detected, the router resolves it using an *incorrect delivery recovery* (IDR) strategy, discussed in §4.2.

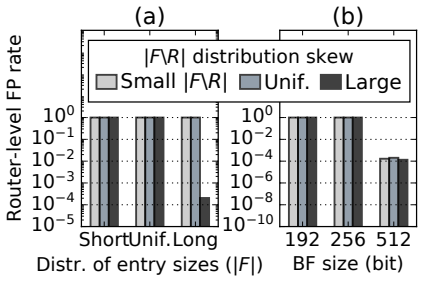

Figure 4: Router-level FP rates for a table with 10^7 entries.

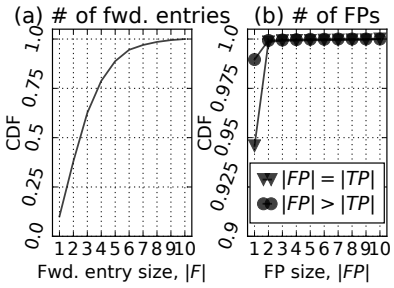

Figure 5: FP matches larger than or equal to TP match. Table size: 2×10^6 entries, BF size: 192 bit.

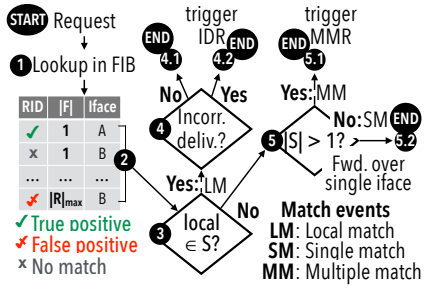

Figure 6: BF-based forwarding algorithm considered in this paper.

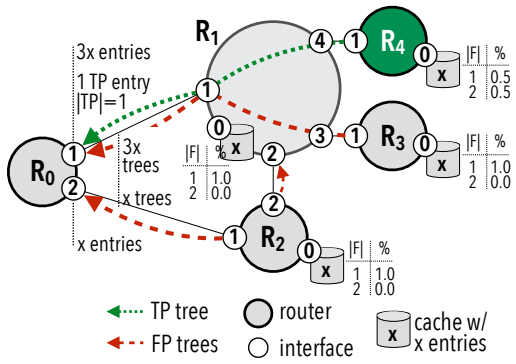

Figure 7: BF-based forwarding model example.

Match events: For ease of presentation, we group different compositions of the interface set S as *match events*, as described in Fig. 6: (1) local match (LM), if local iface $\in S$; (2) single match (SM), if $|S| = 1$; and (3) multiple match (MM), if $|S| > 1$. If step 5 in Fig. 6 results in a SM event, the request is simply forwarded over the single interface in S. In case of a MM or LM event, a MMR or IDR strategy is triggered, respectively.

4.2 Resolution & Recovery Strategies

We now describe the strategies used for resolution of multiple matches and recovery of incorrect deliveries.

Multiple Match Resolution (MMR) strategies: An MMR strategy is triggered if the cardinality of the interface set is larger than 1. In other words, MMR is used when FP matches are detected. For the remainder of this paper, we consider the following MMR strategies:

(1) **All matched links (AML):** Forward the request over all interfaces in the interface set S. If FP rates are high, this strategy approximates the behavior of a flooding forwarding algorithm, thus resulting in high link usage.

(2) **'Fallbacks':** Forward the request using a 'fallback' address, a mechanism introduced by the eXpressive Internet Architecture (XIA) [6]. Fallbacks allow a requester to explicitly specify an alternative address to reach the final destination. E.g., the IP address of a content server as an alternative to an RID. This can reduce the link usage in the network and number of incorrect deliveries relative to the AML strategy, specially if FP rates are high.

Incorrect Delivery recovery (IDR) strategies: An IDR strategy is triggered if an LM event is detected and the request has been delivered to an incorrect content source. Incorrect deliveries can be determined by having caches inspect content metadata, e.g., names in string format. For the remainder of this paper, we consider the following IDR strategies:

(1) **Feedback:** Send the request back to the source, which should then use an alternative addressing method to request the content of interest.

(2) **'Fallbacks':** Similarly to the use of fallbacks in MMR, the requester can set a fallback address to resolve incorrect delivery situations. This can reduce the latency penalty of the feedback strategy, since recovery is handled by the network, without direct intervention of the requester

5 BF-BASED NETWORK MODEL

We present a model which describes the flow of a BF-based request over an arbitrary network topology. The model follows a Bayesian Network approach to calculate metrics which describe the effects of False Positive (FP) matches: e.g., the expected value of match events and request outcomes of different types (e.g., correct and incorrect deliveries). At its core, the model considers the BF-based forwarding engine described in §4, and uses the False Positive (FP) rate expressions given in §3.2 to capture the adverse impact of FP matches (as described in §3.1).

With such a model, we can evaluate the feasibility of BF-based forwarding, without a full ICN architecture implementation or its adaptation to a complex network simulation tool such as ns-3 [16]. An implementation of this model is freely available online[1] (used to conduct the evaluations in this paper).

5.1 Overview

We now provide an overview of our model. To illustrate our description, we use a running example depicted in Fig. 7.

Inputs: The input parameters consist in (1) a description of a network topology (top of Table 2); and (2) the BF parameters used in requests and forwarding entries (bottom of Table 2). The parameters of type (1) are used to model the flow of a request over an arbitrary network topology, and determine the correctness of

[1] https://github.com/adamiaonr/rid-analytics

Parameter	Symbol	Used to/for
Network topology graph	-	Request flow
Trees of size f reachable via iface i	$B[i][f]$	Prob. of 'wrong tree bindings' Flow correctness.
True positive sizes per iface	$\|TP\|_i$	Prob. of $\|TP\|$-sized TP match on iface i.
Max. request size	$\|R\|_{max}$	
Req.-entry diff. per iface ($\|F \backslash R\|$)	$\Phi_i(f)$	Calculate prob. of
Entry sizes per iface	$\Psi_i(f)$	FP match of size
Table size (# of entries)	-	f on iface i.
# of entries per iface	-	

Table 2: Summary of model parameters.

BF-based forwarding operations. Those of type (2) are concerned with the calculation of FP match probabilities at any given router.

In Fig. 7, the topology has 4 routers and 5 links. Each router contains x entries in their caches: assuming each router announces all of their content to every other router, all routers have a forwarding table with $5x$ entries. A request is assumed to start at R_0. A total of $4x$ trees are announced to R_0: $3x$ trees over interface 1, and x over interface 2. Furthermore, since R_4 holds entries of 2 different sizes - 1 and 2 - R_0's interface 1 receives $0.5x$ trees of size 2 and $2.5x$ trees of size 1. Router R_4 holds the requested content, with size $|TP| = 1$, which is announced to R_0 via interface 1. As such, R_0's interface 1 has a TP entry of size 1.

Outputs: The model outputs three different types of values: (1) **expected value of match events** after each forwarding operation, at any given router of a topology; (2) the **expected value of request outcomes** of different types; and (3) the **average latency** associated with each request outcome, expressed in number of traversed hops.

Match events have been introduced in §4. In §5.2, we briefly describe how we model the flow of requests over a topology, and determine the expected value of match events, outcomes and average latency. Due to space limitations, we provide a detailed explanation of the probabilities underlying the calculation of the output metrics in Appendix A.

5.2 Flow of BF-based Requests

When a request arrives at some router R, it is subjected to a *forwarding decision*, i.e. the forwarding engine picks one (or more) interface(s) over which to forward the request. A sequence of consecutive forwarding decisions at different routers represents the *path* of a packet over some network topology. E.g., in Fig 7, all possible paths start at R_0, and may terminate at any of the routers in the topology. Possible paths are $[R_0 \rightarrow R_1]$, $[R_0 \rightarrow R_1 \rightarrow R_3]$ or $[R_0 \rightarrow R_1 \rightarrow R_2 \rightarrow R_0]$. Each of these paths has a probability associated with it.

5.2.1 Outcomes types. Outcomes represent the 'fate' of a request packet at the end of a path. Essentially, an outcome depends on (1) the path followed by a request; and (2) the 'correctness' of the path at its termination, i.e. if the path ends at a node which holds (or does not hold) the requested content. The former case is referred to as a **correct delivery (CD)**, the latter as an **incorrect delivery (ID)**. E.g., in Fig 7 the path $[R_0 \rightarrow R_1 \rightarrow R_3]$ results in a incorrect delivery outcome, while the path $[R_0 \rightarrow R_1 \rightarrow R_4]$ results in a correct delivery. The probability of an outcome is that of the associated path.

Other intermediate outcomes are considered when 'fallback' MMR or IDR strategies are in place, namely having a request follow a fallback address after an MM event and/or an incorrect delivery outcome. However, for the remainder of this paper, we only consider the probability of correct and incorrect delivery outcomes.

5.2.2 Path construction and output generation. We now describe the construction of paths and calculation of output values, using the topology in Fig 7. We assume a request is issued in router R_0, and that routers follow the 'All Matched Links' (AML) multiple match resolution strategy.

Step 1: Starting at R_0, the set of all possible paths is defined as $\{[R_0]\}$, with probability 1, i.e. $P([R_0]) = 1$.

Step 2: Using the BF input parameters defined for R_0 (top of Table 2), we calculate the probability of different forwarding decisions at R_0, $P(i|R_0)$, for all $i \in \{1, 2\}$. This represents the probability of having a packet forwarded over interface i at R_0. The calculation of $P(i|R_0)$ is derived from the basic expressions in §3. Details on their calculation are given in Appendix A. Also, note that we do not consider interface 0 in R_0. This is due to the following reasons:

- At any router R, we do not allow forwarding over the ingress interface, i.e. $P(i_{\text{ingress}}|R) = 0$
- By convention, we consider that $i_{\text{ingress}} = 0$ for the initial router.

Step 3: With $P(i|R_0)$, we can now update the paths and their probabilities:

$$P([R_0 \rightarrow R_1]) = P(1|R_0)\,P([R_0]) \qquad P([R_0 \rightarrow R_2]) = P(2|R_0)\,P([R_0])$$

The set of all possible paths thus becomes $\{[R_0 \rightarrow R_1], [R_0 \rightarrow R_2]\}$. Furthermore, we calculate the expected value of match events at R_0:

$$E[\text{LM}] = P_{\text{I}}(0|R_0)\,P([R_0])$$

$$E[\text{SM}] = \sum^{\forall i} P_{\text{E}}(i|R_0)\,P([R_0])$$

$$E[\text{MM}] = \sum^{\forall i} P_{\text{I}}(i|R_0)\,P([R_0])$$

The 'E' and 'I' subscripts in the forwarding decision probabilities above refer to 'exclusive' and 'inclusive'. $P_{\text{E}}(i|R_n)$ represents the probability of exclusively choosing interface i: i.e. no other interface $j \neq i$ is chosen. $P_{\text{I}}(i|R_n)$ represents the probability of choosing interface i, regardless of the simultaneous choice of another interface $j \neq i$. At R_0, $E[MM]$ can be at most 2, if all egress interfaces are chosen simultaneously, leading the use of 2 links. By definition (given in §4), $E[LM]$ and $E[SM]$ can be at most 1.

Step 4: We repeat steps 2 and 3 until all possible paths terminate. A path is terminated when: (1) the request is delivered to the local interface at some router R (interface 0, by convention); or (2) its length exceeds a given limit. (2) is equivalent to a 'time-to-live' (TTL) field, typically set to the diameter of the topology. TTLs are a standard approach to deal with loops in the Internet [4], and using the network diameter as an initial value is aligned with common practices [4]. E.g., in Fig 7 the path $[R_0 \rightarrow R_1 \rightarrow R_2 \rightarrow R_0]$ would not be possible, since it exceeds the topology diameter, 2 hops.

Step 5: At the end of step 4, we have all possible paths which can followed by a request in the topology, associated with a probability,

AS	# nodes	# links	Outdegree		Path lengths	
			Mean	Max.	Mean	Max.
1221, Telstra (AU)	44	44	2.0	18	3	6
3257, Tiscali (Europe)	41	87	4.2	28	2.5	5
3356, Level3 (USA)	62	284	9.2	39	2	4
7018, AT&T (US)	114	148	2.6	25	3	6

Table 3: Characteristics of Rocketfuel PoP-level topologies used in experiments.

and a latency τ. We then determine the outcomes for every path, by determining their correctness. E.g.:

$$P([R_0 \rightarrow R_1 \rightarrow R_3]) = P(3|R_1)\,P([R_0 \rightarrow R_1]),\ \tau = 2 \text{ hops, CD}$$

$$P([R_0 \rightarrow R_1 \rightarrow R_4]) = P(4|R_1)\,P([R_0 \rightarrow R_1]),\ \tau = 2 \text{ hops, ID}$$

As a final step, we calculate the expected values of the outcomes and latencies, for each outcome type. E.g., considering the set of paths terminating in an incorrect delivery (ID) outcome - $\mathcal{S} = \{[R_0 \rightarrow R_1], [R_0 \rightarrow R_1 \rightarrow R_3], [R_0 \rightarrow R_1 \rightarrow R_2], [R_0 \rightarrow R_2], [R_0 \rightarrow R_2 \rightarrow R_1]\}$ - we calculate these values as follows:

$$E[\text{ID}] = \sum^{\forall s \in \mathcal{S}} P(s) \qquad E[\text{ID } \tau] = \frac{\sum^{\forall s \in \mathcal{S}} P(s)\,\tau(s)}{E[\text{ID}]}$$

6 EVALUATION

We now evaluate the feasibility of Bloom filter (BF) based forwarding in a network context. In particular, we answer three questions: (1) *what is the impact of FP matches in a network-wide context?*; (2) *how do these effects change for different network scenarios?*; (3) *how do the multiple match resolution (MMR) and incorrect delivery recovery (IDR) strategies introduced in §4 affect the basic forwarding algorithm beneficial?* We use the analytical model presented in §5 to evaluate BF-based forwarding over realistic PoP-level topologies from Rocketfuel [20]. We use three broad categories of metrics in our evaluations:

Link usage: The number of links over which request packets are forwarded after a forwarding decision. This is either measured in absolute or relative numbers: the latter compares the number of links used in BF-based forwarding to that used by any form of ICN forwarding/routing which follows exact match semantics and shortest path routing.

Forwarding correctness: The relative share of match events measured in a particular scenario, namely the percentage of correct and incorrect forwarding decisions, along with the share of multiple match events (as defined in §4.1).

Latency: The number of hops traversed by a correct request delivery, in any arbitrary topology.

6.1 Evaluation Setup

We now present the network topologies we use in our evaluations, and the evaluation parameters and use cases.

6.1.1 Network Topologies. Table 3 summarizes the main characteristics of the PoP-level topologies we use in our experiments. These have been adapted from PoP-level topologies in the Rocketfuel dataset [20] and provide realistic values for the parameters which affect link usage and latency: (1) *length* of paths (in number of hops); (2) *node outdegree* (or equivalently, the number of interfaces in a router); and (3) number of nodes in the topology (alternatively, number of routers). The more hops a request traverses, the higher the likelihood of a forwarding error. Table 3 shows paths as long as 6 hops, which is consistent with average inter-domain path lengths in the current Internet (~ 4 hops [2]). The combination of node outdegree and number of nodes influences link usage, since nodes with large outdegrees may forward FP matches over many links. As shown in Table 3 we consider topologies with small mean outdegrees - e.g., Telstra (1221) and AT&T (7018) - as well as more complex topologies - e.g., Level3 (3356) - with mean outdegrees of 9.2 and and a maximum outdegree of 39.

6.1.2 Content Distribution. We assume that a space of P prefixes (or alternatively, forwarding entries) is announced within a topology of N nodes. Each topology node is a source for an equal share of the the prefix space, i.e. P/N prefixes. Similar assumptions are made in [3], which considers PoP-level topologies as core networks, each node associated with an access network. We assume nodes announce their prefixes to every other node, via shortest path spanning trees, as discussed in §2.2. Shortest paths are determined by the number of hops. Note that in the PoP-level topologies we consider, nodes correspond to domains. We assume each domain coordinates its caches internally, and as such the 'domain cache' can be viewed as a single integrated cache.

6.1.3 Evaluation Parameters. Table 4 lists the parameters used in our experiments, organized into 3 scenarios: (1) global reachability, (2) CDN-style (or 'pro-active') caching, and (3) opportunistic (or 'reactive') caching. We describe the scenario details in §6.2, and discuss general parameter choices below.

Bloom filter sizes: We start with BF sizes of 192 bit, as proposed in earlier work [12, 13]). To evaluate the effect of larger BF sizes, we also consider 256, 384 and 512 bit.

Request sizes: RID requests encode up to a maximum of 10 inner prefixes. This falls within the ranges used in previous work [12, 13]. In some experiments we set request sizes between 3 and 10 to test the effect of prefix exclusion.

Request-entry differences per link: In all experiments, we use large request-entry differences ($|F \backslash R|$) on forwarding tables, on all interfaces. More specifically, we assume that over 90% of the entries on an interface exhibit one of the 2 largest $|F \backslash R|$ values. E.g., > 90% of the entries have $|F \backslash R|$ equal to 9 or 10, when $|R|_{max} = 10$. Also, we assume at least one entry per $|F \backslash R|$ value. As seen in §3.2.2, large $|F \backslash R|$ values are the most favorable for BF-based forwarding.

Forwarding table sizes: Forwarding tables are generally set to size 10^7, a value which is consistent with recent ICN research [13, 22]. In some evaluation cases, we consider other powers of 10.

6.1.4 Methodology. Our experiments follow 4 steps:

Step 1: For each topology, we pick 20 different source and destination node pairs, chosen at random, provided that the path length in-between the nodes is as specified by the scenario. We do this to get a diverse set of paths, so that the results are not biased towards a specific type of path (e.g., paths starting at nodes with low outdegree).

Scenario	Topologies	Req. sizes	Forwarding entries								Table size	BF sizes	MMR	
			Origin servers				Caches						AML	FB
			#	$\|F\|$	$\|TP\|$	# hops	#	$\|F\|$	$\|TP\|$	# hops				
1) Global reach.	All 3356, 7018	10 [3, 10]	1	{1, 5}	{1, 5}	4	0	-	-	-	10^7 [10^5, 10^8]	{192, 256, 384, 512} 192	✓	-
2) CDN caching	3356, 7018	10	1	1	1	4	$\{n/2, n\}$	{4, 5}	- 5	2	10^7	384 192 384 192	✓ - ✓ -	- ✓ - ✓
2) Opport. cach.	3356, 7018	10	1	1	1	4	$\{n/2, n\}$	{7, 10}	- 7	2	10^7	384 192 384 192	✓ - ✓ -	- ✓ - ✓

Table 4: Summary of evaluation scenarios & parameters.

Step 2: For each pair obtained in step 1, we generate several evaluation cases, one for each possible combination of the remaining evaluation parameters.

Step 3: We run the model for each evaluation case and collect the outputs. We do not repeat experiments for a particular evaluation case, since the outputs of the model are deterministic for a particular case. The diversity of results within a topology comes from the set of paths chosen in step 1.

Step 4: Finally, for each topology and parameter combination, we calculate the average values of the outputs over the 20 different source and destination pairs.

6.2 Results and Discussion

6.2.1 Scenario 1: Global Reachability. In scenario 1, we evaluate BF-based forwarding when TP sources are relatively far from requesters and aggregation is high (i.e., announcements are short, $|F| = 1$ and $|F| = 5$). We set the path length towards the origin server to 4 hops, which is the maximum common longest path to all the topologies, and consistent with path lengths observed in the current Internet [2].

Link usage: Fig. 8 shows the average number of links which are used in the network when a single request is cast at a random router. We assume the 'All Matched Links' (AML) MMR strategy is in effect (see §4.2). Under aggressive aggregation ($|F| = 1$) and short BF sizes (192 and 256 bit), AML approximates the behavior of a flooding algorithm: the replication of packets over multiple egress interfaces is significant at every router, resulting in high link usage. Only 512 bit BFs reach the ideal link usage. Increasing entry sizes to $|F| = 5$ makes 384 bit BFs reach the ideal link usage (4 links), but 256 bit BFs can still result in an excess, which is aggravated by the complexity of the topology. E.g., for Level3 (3356), the excess goes over 5. This result shows a trade-off between aggregation levels and forwarding correctness. Note that an increase in forwarding entry size would imply aggregation loss, and thus larger forwarding tables.

Prefix exclusion & table sizes: Fig. 9 plots the link usage excess when varying table sizes and request sizes. In this case, the excess is relative to the ideal usage of 4 links, from request source to the origin server. We use 192 bit BFs, again considering short entries ($|F| = 1$). The benefits of prefix exclusion are clear: if request sizes are reduced to 3 or 4, there is virtually no link usage excess, even for table sizes as large as 10^8. On the other hand, Fig. 9 shows that even if table sizes drop by two orders of magnitude (from 10^7 to 10^5), the link usage excess without prefix exclusion (request size 10) remains large for the larger topologies (e.g., Level3 and AT&T).

These results show that prefix exclusion may be a good approach to improve BF-based forwarding correctness with shorter BF sizes, as evidenced by the lower link usage values. However, prefix exclusion may imply larger forwarding tables, which increases FP rates. As such, a definitive conclusion about the effectiveness of prefix exclusion can only be taken after a thorough analysis of its impact on table sizes.

Takeaways:

- *Our link usage analysis of 'vanilla' BF-based forwarding over realistic PoP-level topologies shows evidence of a trade-off between aggregation and forwarding correctness.*
- *Prefix exclusion improves forwarding correctness. However, since prefix exclusion implies larger forwarding tables, a definitive conclusion requires a thorough analysis of a trade-off between prefix exclusion and table sizes.*

6.2.2 Scenario 2: Pro-active caching. In scenario 2, we extend scenario 1 with mid-sized entries - $|F| = \{4, 5\}$ - announced by 2-hop neighbors of the request source. We henceforth refer to such network nodes as 'caches'. We also vary the number of caches around the request source: namely, (1) half of the 2-hop neighbors acting as caches (case '$n/2$' in Table 4); (2) all neighbors acting as caches (case 'n').

This scenario serves multiple purposes. First, we study BF-based forwarding under LPM, since we now have different entry sizes in forwarding tables. Second, we look at how varying the number of caches impacts multiple matches and forwarding correctness. Finally, we compare the performance of 'fallback' multiple match resolution to that of 'vanilla' BF-based forwarding. Fallback multiple match resolution is evaluated with 192 bit BFs, and compared against the AML strategy with 384 (2×192) bit BFs. We argue this is a fair comparison, since fallbacks imply the use of additional addressing information. We compare performance according to (1) link usage excess; and (2) latency. We divide scenario 2 into two sub-cases, according to the presence (or absence) of True Positive (TP) entries pointing towards cache nodes.

Caches with no TP entries: Fig. 10(a) shows that AML forwarding performs adequately when 384 bit BFs are used, even for large topologies such as AT&T and Level3. The link usage excess is low,

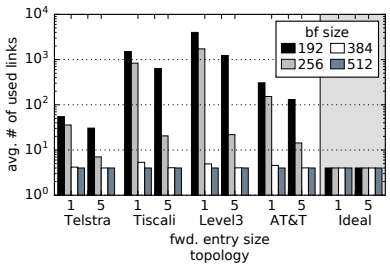

Figure 8: Results for scenario 1.1.

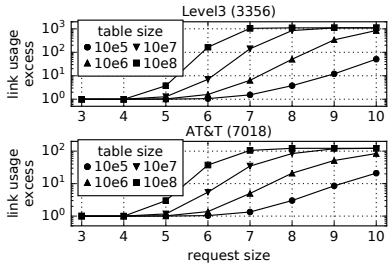

Figure 9: Results for scenario 1.2.

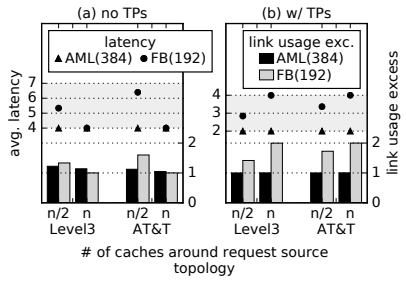

Figure 10: Results for scenario 2.

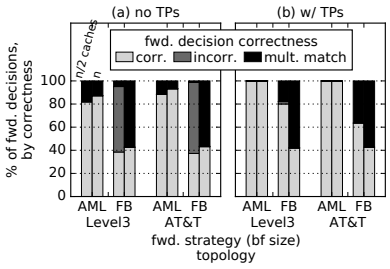

Figure 11: Results for scenario 2 (cont.).

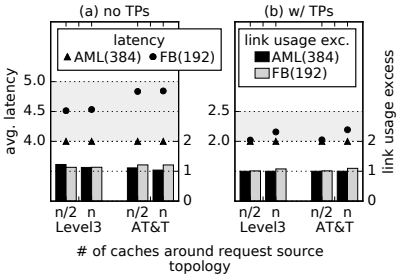

Figure 12: Results for scenario 3.

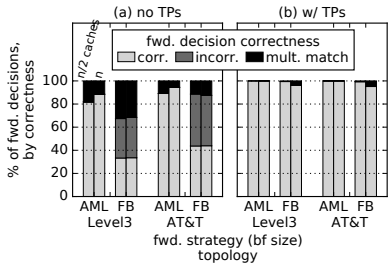

Figure 13: Results for scenario 3 (cont.).

and the average delivery latency is the ideal of 4 hops, the distance towards the origin server. Fig. 11 shows that with 384 bit BFs, the occurrence of incorrect forwarding decisions is negligible: decisions are either correct or resolved via 'flooding' after FP detection, as evidenced by the occurrence of a small percentage of multiple matches.

For the 192 bit & fallback combination, results are ideal when all 2-hop neighbors of the request source act as caches. This is evidenced by an ideal link usage excess of ~ 1, and average latency of 4 hops. However, performance deteriorates when only a portion of the 2-hop neighbors act as caches, due to incorrect forwarding decisions, as shown in Fig. 11. FP rates with 192 bit BFs are high, even for mid-sized entries such as $|F| = \{4, 5\}$. Since cache announcements are larger than origin server announcements, the likelihood of FP matches on the interfaces pointing towards caches increases. Thus, the LPM forwarding engine deviates the request from its TP path. When all 2-hop neighbors act as caches (case n), multiple matches are more likely, thus triggering FP detection and subsequent resolution via fallbacks.

Caches with TP entries: Fig. 10(b) shows that AML also works well with 384 bit BFs when caches hold the requested content: delivery latencies are ideal - 2 hops towards the cache - and link usage excess is negligible.

However, the combination of 192 bit BFs & fallbacks still delivers the content to the origin server, and not a cache. This is shown by large latencies and excess link usage in Fig. 10(b). Again, this is due to the triggering of fallback resolution after the first hop, which immediately sends the request towards the origin server. Fallbacks are triggered by multiple matches at the beginning of the path, since all 2-hop caches announce entries of the same size. Fig. 11 shows that multiple matches become more likely as the number of caches increases. For some paths picked at random, the number of 2-hop neighbors is small, such that the $n/2$ case considers a single 2-hop cache. Since this cache also announces the TP entry, the latency towards the content becomes 2 hops. This is why the latency of the $n/2$ case is lower than 4 hops, on average.

Takeaways:

- *Longest Prefix Matching (LPM) forwarding semantics are correctly followed by BF-based forwarding for sufficiently large BF sizes, e.g, 384 bit, larger than those proposed by previous work [12, 13].*
- *The use of 'fallback' addresses is only useful when FP detection is very likely, which may only happen in a limited number of cases. E.g., FP detection is likely when all neighbors of a request source announce mid-ranged prefixes, but unlikely if only a subset of the neighbors do it.*

6.2.3 Scenario 3: Opportunistic caching. We use the same setup as scenario 2, only with larger announcements from the caches located 2 hops from the request source. We consider announcements of size 7 and 10, which represent the case of 'reactive' (or opportunistic) caching of content frequently requested from the access networks associated with a PoP-level node. Fig. 12 shows average latencies for two arrangements: (a) no TP entries announced by caches; (b) a TP entry (size 7, smaller than 10) is announced by a 2-hop cache.

The results for AML strategy are the same as in scenario 2: this is expected, due to the low FP rates caused by the relatively large BF sizes (384 bit) and large size of cache announcements. In fact, Fig. 13 shows the same distribution of forwarding decision types as Fig. 11 for the AML strategy.

The 192 bit BF & fallback combination shows near ideal link usage, but a latency overhead when caches do not hold the requested content (Fig. 12(a)). This is due to the penalty of incorrect deliveries, which occur often, as shown in Fig. 13. In this scenario, the LPM engine only sees matches of size $|F| = 1$ on multiple interfaces -

including the local interface - because FP rates are low among the large entries announced by caches (sizes 7 and 10). Since the forwarding engine prioritizes local matches (§4), it incorrectly delivers the request locally. Then, it resolves the incorrect delivery with a fallback.

Despite the smaller size of the TP entry (size 7) when compared to the announcements of other caches (size 10), the request is correctly forwarded in both AML and fallback forwarding strategies. Again, this is dud to the low FP rates among large entries.

Takeaways:

- *Forwarding errors caused by opportunistic cache announcements are rare, provided that announcement sizes are large (e.g., size 10).*
- *Multiple match resolution strategies do not provide visible performance benefits vs. the use of larger BF sizes.*

7 RELATED WORK

Forwarding with Bloom Filters: In an ICN context, the use of BFs in packet forwarding has been proposed in the past [9, 12–14, 19]. In [14, 19], packets carry NDN-style names, and BFs are only used in forwarding tables. The goal is to attain space efficiency in tables, by having BFs aggregate NDN-style prefixes announced over the same interface. This allows routers to reduce false positive (FP) rates independently, through the adjustment of BF parameters. In our case, BFs are used in both request and data packets, and its parameters must be consistent among all network entities, which precludes independent FP rate control at each router. In an ICN context, LIPSIN [9] proposes a novel multicast fabric based on BFs, useful for publish/subscribe applications. BFs encode the set of links in a multicast tree, which is carried by packets. When a packet arrives at a router, it is forwarded over the interface IDs which match those in the BF. BFs are built by a network service with access to the network topology and subscribers in a multicast group. This provides an higher degree of control over FP rates than in our case, since we require clients to independently build the BFs carried by packets. To handle FPs, LIPSIN proposes the use of multiple IDs for the same link (alternative forwarding state, as with fallbacks), or feedback mechanisms which enable routers to ignore certain BFs. Tsilopoulos et al. [21] use BFs in a similar way to LIPSIN to reduce the amount of forwarding state information kept by routers in an NDN network. Papalini et al. [12, 13] propose the usage of BFs as the network representation of content descriptors (i.e., sets of string tags) to achieve routing scalability. The name encoding and lookup methods presented in §2 follow from this work. The authors show the scalability benefits of this work, but overlook the impact of FPs on forwarding correctness. Our work aims at filling this knowledge gap.

Performance of BF-based forwarding: The descriptions given in §2 assume simplified abstractions of the notions forwarding tables and routers. Other work provides more detail about these concepts: e.g., in [12, 13], forwarding tables are organized as PATRICIA tries. However, we note that such details are useful to optimize lookup performance, but do not help reduce the negative impact of False Positive (FP) matches. Since our goal is to study the impact of FP matches and not the performance of BF-based forwarding, we opted to maintain our simplified version of forwarding tables and lookup procedures.

Fallbacks: In an non-ICN context, Yu et al. [23] propose a method which uses BFs in both packets and forwarding tables, without intervention of network addresses. To handle FP matches, the authors propose the concept of 'redirection', which resembles that of 'fallbacks'. The redirection target is chosen by the control plane, without intervention of endpoints. This contrasts with fallbacks as used in XIA [6], which allows endpoints to define custom redirection targets.

Feasibility of BF-based forwarding: Katsaros et al. [10] analyze the impact of using BFs to realize inter-domain name resolution in ICN architectures. Through simulations of inter-domain topologies, the authors find that the overall FP rates experienced at an inter-domain level are large, deeming the scheme impractical. Similarly to our case, the authors witness the severe impact of routing tables composed by large numbers of individual BFs. This study notes that in a hierarchical network architecture like DONA [11], the overall FP rate increases at every hop: this follows routing tables that aggregate those of lower levels, thus resulting in a cumulative increase of FP rate.

8 CONCLUSIONS

In this paper we analyze the correctness of Bloom filter (BF) based forwarding, as recently proposed in an ICN context [12, 13]. We find that the occurrence of forwarding errors increases when aggregation in forwarding tables is high, identifying a trade-off which puts forwarding correctness at odds with table scalability. Our evaluations over realistic PoP-level topologies show that BF-based forwarding results in excessive link usage, using BF and forwarding table sizes proposed in ICN literature [12, 13, 22]. We find that increasing BF sizes from the proposed 192 bit to 384 bit improves the performance of BF-based forwarding to nearly ideal levels.

We propose two techniques - prefix exclusion and 'fallbacks' - to reduce FP rates at routers and mitigate the negative impact of FP matches, respectively. Our evaluation shows that prefix exclusion shows great promise to reduce network-wide FP rates. However, a definitive conclusion requires a thorough analysis of the impact of prefix exclusion on forwarding table sizes: an increase in table sizes may offset the benefits of prefix exclusion. We show that fallbacks can be useful when FP rates are high, but detrimental otherwise, specially when compared to the use of a an equivalent BF size (e.g., 384 bit). Further analysis of network topologies and application namespaces is required to assess the prevalence of good network conditions for fallbacks.

REFERENCES

[1] A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. In *Internet Mathematics*, pages 636–646, 2002.

[2] Y.-C. Chiu, B. Schlinker, A. B. Radhakrishnan, E. Katz-Bassett, and R. Govindan. Are We One Hop Away from a Better Internet? In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, IMC '15, pages 523–529, New York, NY, USA, 2015. ACM.

[3] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable icn. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 147–158, New York, NY, USA, 2013. ACM.

[4] I. E. T. Force. Requirements for Internet Hosts, Communication Layers. Internet Requests for Comments, October 1989.

[5] N. Fotiou, D. Trossen, and G. C. Polyzos. Illustrating a Publish-Subscribe Internet Architecture. *Telecommunication Systems*, 51(4):233–245, 2011.

[6] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: Efficient Support for Evolvable Internetworking. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, page 23, Berkeley, CA, USA, 2012. USENIX Association.

[7] 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*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[8] Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. The web09-bst Dataset. `http://boston.lti.cs.cmu.edu/Data/web08-bst/planning.html`.

[9] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. Lipsin: Line speed publish/subscribe inter-networking. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 195–206, New York, NY, USA, 2009. ACM.

[10] K. V. Katsaros, W. K. Chai, and G. Pavlou. Bloom Filter Based Inter-Domain Name Resolution: A Feasibility Study. In *Proceedings of the 2Nd International Conference on Information-Centric Networking*, ICN '15, pages 39–48, New York, NY, USA, 2015. ACM.

[11] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-oriented (and Beyond) Network Architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pages 181–192, New York, NY, USA, 2007. ACM.

[12] M. Papalini, A. Carzaniga, K. Khazaei, and A. L. Wolf. Scalable Routing for Tag-based Information-centric Networking. In *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, pages 17–26, New York, NY, USA, 2014. ACM.

[13] M. Papalini, K. Khazaei, A. Carzaniga, and D. Rogora. High throughput forwarding for icn with descriptors and locators. In *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, ANCS '16, pages 43–54, New York, NY, USA, 2016. ACM.

[14] D. Perino and M. Varvello. A Reality Check for Content Centric Networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 44–49, New York, NY, USA, 2011. ACM.

[15] D. Perino, M. Varvello, L. Linguaglossa, R. Laufer, and R. Boislaigue. Caesar: A content router for high-speed forwarding on content names. In *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '14, pages 137–148, New York, NY, USA, 2014. ACM.

[16] G. F. Riley and T. R. Henderson. *The ns-3 Network Simulator*, pages 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[17] A. Rodrigues, P. Steenkiste, and A. Aguiar. Analysis and Improvement of Name-based Forwarding over Flat ID Net. Arch. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ICN'18, New York, NY, USA, 2018. ACM.

[18] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav. A Global Name Service for a Highly Mobile Internetwork. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 247–258, New York, NY, USA, 2014. ACM.

[19] W. So, A. Narayanan, and D. Oran. Named data networking on a router: Fast and dos-resistant forwarding with hash tables. In *Proceedings of the Ninth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '13, pages 215–226, Piscataway, NJ, USA, 2013. IEEE Press.

[20] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2–16, Feb. 2004.

[21] C. Tsilopoulos, G. Xylomenos, and Y. Thomas. Reducing forwarding state in content-centric networks with semi-stateless forwarding. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 2067–2075, April 2014.

[22] Y. Wang, Y. Zu, T. Zhang, K. Peng, Q. Dong, B. Liu, W. Meng, H. Dai, X. Tian, Z. Xu, H. Wu, and D. Yang. Wire speed name lookup: A gpu-based approach. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 199–212, Berkeley, CA, USA, 2013. USENIX Association.

[23] M. Yu and J. Rexford. Hash, Don'T Cache: Fast Packet Forwarding for Enterprise Edge Routers. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, pages 37–44, New York, NY, USA, 2009. ACM.

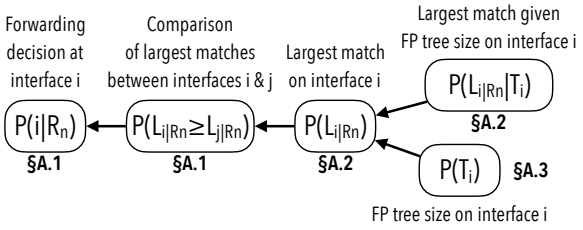

Figure 14: Bayesian network used for $P(i|R_n)$ calculation.

A PROBABILITY FRAMEWORK

We now explain the details behind the calculation of the outputs in our model. Both the expected value of match events and request outcomes are derived from one essential value: the probability of a forwarding decision at router R_n, $P(i|R_n)$. A forwarding decision corresponds to the event '*forwarding a request over interface i at router R_n*'.

We follow a Bayesian Network approach for the calculation of $P(i|R_n)$, as depicted in Fig. 14. We dedicate a brief subsection to each the probabilities involved in the network.

A.1 Probability of Forwarding Decisions

$P(i|R_n)$ depends on the comparison of 'largest matches' among all the interfaces of a router. More specifically, interface i is chosen if the following event occurs: '*the size of i's largest match is larger than (or equal to) that of the largest match at any other interface $j \neq i$*'. We represent the size of the largest match on interface i using the random variable (RV) $L_{i|R_n}$. The size of a largest match can range from 0 (i.e., having no match at all) to $|R|_{max}$. Using this notation, the probability of deciding for interface i follows as $P(i|R_n) = P(L_{i|R_n} \geq L_{j|R_n}), \forall j \neq i$.

To get $P(L_{i|R_n} \geq L_{j|R_n}), \forall j \neq i$, we first determine the values of largest match probabilities per interface: $P(L_{i|R_n})$ and $P(L_{j|R_n})$. Then, we calculate $P(L_{i|R_n} \geq L_{j|R_n})$ via a joint probability, as shown in Eq. 6:

$$P(i|R_n) = P(L_{i|R_n} \geq L_{j|R_n}) = \sum_{l=1}^{|R|_{max}} \sum_{k=1}^{K} P(L_{i|R_n} = l)\, P(L_{j|R_n} = k) \tag{6}$$

We assume the events $L_{i|R_n} = l$ and $L_{j|R_n} = k$ are independent. This is a reasonable assumption, since the occurrence of each event only depends on the input parameters associated with the respective interfaces i and j.

$P(i|R_n)$ can either be calculated as $P(L_{i|R_n} > L_{j|R_n})$ or $P(L_{i|R_n} \geq L_{j|R_n})$, depending on the value of K in Eq. 6. The former is used to calculate the probability of a Single Match (SM) event at interface i, using $K = l - 1$: interface i is chosen iff its largest match is strictly larger than that at any other interface. The latter (using $K = l$) is used to calculate the probability of a Multiple Match (MM) event at interface i.

A.2 Largest Match Probabilities

The probability of largest matches - $P(L_{i|R_n} = l)$ - is independently calculated per interface. Its value depends on the probability of different match types which can occur at an interface: (1) true positive (TP) matches; and (2) false positive (FP) matches. Below we explain the contribution of each match type.

TP matches: The contribution of TP matches to $P(L_{i|R_n})$ directly depends on the largest TP sizes at interface i, $|TP|_i$. If $|TP|_i > 0$, then we are guaranteed to have a match of size $|TP|_i$ at interface i. However, $|TP|_i$ becomes the largest match iff there are no FP matches larger than $|TP|_i$. In addition, $|TP|_i$ makes the occurrence of largest matches of an inferior size impossible, i.e. $P(L_{i|R_n} < |TP|_i) = 0$.

FP matches: In our model, a FP match can be explained in one of two ways: (1) as a 'first-time' FP match; or (2) a match caused by a 'binding' of the request to a FP routing tree, which announces entries of size t [2]. The former implies the request is not yet bound to a FP tree. Topology descriptions provide a rough differentiation between entries which are 'fresh' or included in FP trees, via source tree bitmasks (see Table 2). The latter is due to a 'first-time' FP match at a previous router.

To accommodate case (2), we introduce the RV '*size of the largest FP tree size the request is bound to at interface i*', T_i. We then calculate largest match probabilities conditioned on the the events $T_i = t$. As such, the conditional probabilities $P(L_{i|R_n}|T_i = t)$ become as in Eq. 7:

$$P(L_{i|R_n} = l|T_i = t) = \begin{cases} 0 & l < max(|TP|_i, t) \\ P(|FP| \leq l) & l = max(|TP|_i, t) \\ P(|FP| = l)\, P(|FP| \leq l) & l > max(|TP|_i, t) \end{cases} \tag{7}$$

The rationale of Eq. 7 is: (1) if a request is bound to a FP tree of size t, or if a TP match exists in interface i, then the largest match is guaranteed to be at least as large as the maximum between t or $|TP|_i$; (2) largest matches of size equal to t or $|TP|_i$ (whichever is maximum) are possible iff larger FP matches do not occur; (3) largest matches of size $l > max(|TP|_i, t)$ happen iff a FP match of size l occurs, while making sure that FP matches larger than l do not occur simultaneously.

Final expression: Combining the contribution of all match types, the expression for the largest match probabilities becomes as in Eq. 8:

$$P(L_{i|R_n} = l) = \sum_{t=0}^{|R|_{max}} P(L_{i|R_n}|T_i = t)\, P(T_i = t) \tag{8}$$

A.3 FP Tree Probabilities

FP tree probabilities - $P(T_i = t)$ - model the concept of 'wrong tree bindings' introduced in §3.1. The RVs T_i refer to events of the type '*the largest FP tree size the request is bound to at interface i [of some router R_n]*'. In our model, the values of $P(T_i)$ are propagated and updated between routers as the request traverses a topology, from 'previous' to 'next'.

In the network topology shown in Fig. 7, $3x$ source trees are announced by R_1, R_3 and R_4 to the origin of the request, R_0. Furthermore, $0.5x$ trees are of size 2 and $2.5x$ trees of size 1.

Initial FP tree binding: The request is emitted at R_0, and initially it is not bound to any FP tree. As such $P(T_1) = 0$ at R_0. As the request is forwarded from R_0 to R_1, it may bind to any of the $3x$ FP trees announced by the routers upstream. At R_0, the probability

[2] For simplicity, we henceforth refer to such routing trees as 'FP trees of size t'.

of having the request bound to a FP tree as it leaves via interface 1 - $P(T_{out,1}|R_0)$ - is solely dictated by the probability of 'fresh' FP matches at R_0's interface 1. As the request flows from R_0 to R_1, we keep a record of the trees which can be reached via R_0's interface 1. We do this via a bitmask, using the $B[i][f]$ parameter in Table 2: the k-th bit of the bitmask $B[i][f]$ indicates if the node k is reachable (1) or not (0) via a tree of size f. E.g., in Fig. 7, at R_0's interface 1, the bitmask would be $B[1][1] = 01011$, $B[1][2] = 00001$.

FP tree probabilities per interface: The request enters R_1 with probability $P(T_{in} = t|R_1) = P(T_{out,1} = t|R_0)$ of being stuck to a FP tree of size t. We then assign a FP tree probability to each of the egress interfaces of R_1 - $P(T_i|R_1)$ - which will be later used in Eq. 8. This assignment is done by dividing $P(T_{in}|R_1)$ among the egress interfaces of R_1, scaled by the ratio of trees shared between each of the egress interfaces and those in R_0's interface 1. This allows us to quantify the ratio of trees which are transfered in-between routers, i.e. the trees with potential for 'wrong tree bindings'. The calculation of $P(T_i|R_n)$ is given by Eq. 9:

$$P(T_i = t|R_n) = \begin{cases} P(T_{in} = t|R_1) \frac{\#\text{ trees of size } t \text{ at } i}{\sum^{\forall i} \#\text{ trees of size } t \text{ at } i} & \text{for } t > 0 \\ 1 - \sum^{\forall t>0} P(T_i = t|R_n) & \text{for } t = 0 \end{cases} \quad (9)$$

E.g., in Fig. 7:

$$P(T_0 = 1|R_1) = P(T_3 = 1|R_1) = \frac{x}{2.5x} P(T_{in} = 1|R_1) = \frac{2}{5} P(T_{in} = 1|R_1)$$
$$P(T_4 = 1|R_1) = \frac{0.5x}{2.5x} P(T_{in} = 1|R_1) = \frac{1}{5} P(T_{in} = 1|R_1)$$
$$P(T_4 = 2|R_1) = \frac{0.5x}{0.5x} P(T_{in} = 2|R_1) = P(T_{in} = 2|R_1)$$
$$P(T_2 = 0|R_1) = 1$$
$$P(T_2 = 1|R_1) = P(T_2 = 2|R_1) = 0$$

Adding 'fresh' FP match contributions: Finally, as the request leaves R_1, we add the contribution of 'fresh' FP matches on egress interfaces. This accounts for situations in which the request binds to a FP in the middle of its path. This can happen because the request experiences FP matches on egress interface i while also: (1) entering R_1 not bound to any FP tree; (2) entering R_1 bound to a FP tree which continues over some interface j other than i; or (3) interface i does not share any trees with the previous router (e.g., in Fig. 7, R_1's interface 2 does not share any trees with R_0.). Case 1 happens iff a request traverses a link due to a TP match and no FP matches occur at any of the previous routers.

The FP tree probability passed to R_3 would then be given as:

$$\begin{aligned} P(T_{out,3} = t|R_1) = {} & P(T_{in} = 0|R_1)\, P(|FP|_3 = t) \\ & + (1 - P(T_{in} = 0|R_1))\, P(T_3 = 0)\, P(|FP|_3 = t) \\ & + (1 - P(T_{in} = 0|R_1))\, P(T_3 = t) \end{aligned}$$

A.4 Computational complexity

The asymptotic time complexity of all probability computations combined is $O(N\,M^3)$, in which N is the number of interfaces in a router, and M the number of possible match sizes. This value is mostly dictated by the computation of Eq. 6. We use a '1-vs-all' approach to reduce its time complexity from $O(M^N)$ to $O(M^3)$.

'Strawman' approach: The joint probability calculation described in Eq. 6 only considers 2 interfaces. For a router with N interfaces, and considering M possible match sizes, the asymptotic time complexity of Eq. 6's computation would be $O(M^{2N})$. This becomes prohibitively complex in most real-life topologies: e.g., in PoP-level topologies traced via Rocketfuel [20], nodes can have up to 39 edges.

1-vs-all approach: We use a '1-vs-all' approach to reduce the computational complexity of Eq. 6's computation. Instead of comparing the largest match of i against *each and every* other interface $j \neq i$, we compare i against the *aggregate of all* interfaces $j \neq i$. We refer to this aggregate as $\sim i$. The aggregate consists in the union of all interface parameters associated with $j \neq i$, as if it were a single interface. This way, the complexity of Eq. 6 is reduced to $O(M^3)$, regardless of the maximum number of interfaces N, which is acceptable for the purposes of our analysis.