On Popularity-Based Load Balancing in Content Networks

Tomasz Janaszka
Orange Labs Poland
Email: Tomasz.Janaszka@orange.com

Dariusz Bursztynowski
Orange Labs Poland
Email: Dariusz.Bursztynowski@orange.com

Mateusz Dzida
Orange Labs Poland
Email: Dzida.MateuszMD@orange.com

***Abstract*—The paper investigates the use of multipath routing and load balancing in content oriented architectures proposed for the next generation Internet. It discusses how various load balancing methods can influence the efficiency of in-network caching and overall network performance measured as file retrieval time. Proposed approach, called Popularity-Aware Load Balancing, differentiates popular and unpopular content, and applies dedicated balancing principle to each content set. Performance of the proposed approach is validated through developed simulator of the content-centric network. Obtained results prove that popularity-based load balancing applied for multi-path routing patterns can outperform single-path routing patterns in the content networks experiencing flash crowds.**

I. Introduction

The use of multiple paths and associated load balancing are traffic engineering techniques used to achieve efficient utilization of network resources. In this paper, the combination of these two capabilities is referred to as multipath routing. The advantages of using multipath routing are limited to relatively narrow network conditions. In fact, it is the case of failures or local congestions when flow diversification associated with the usage of multipath routing can bring benefits by alleviating the load on some network elements. In this paper, we follow this general assumption.

In the above context, a plethora of strategies have been proposed in the literature to partition flows over sets of admissible paths [1], [2], [3], [4], [5]. However, high-grade network-level multipath routing should handle traffic in a flow-aware manner: one should either to partition/repartition application flows that are sent (split) over distinct paths or assure that each flow is sent over a single path. Both of these options call for smart load balancing mechanisms/algorithms to become operationally viable and none of them have received strong support in standardization (we notice that the best known Equal-Cost Multi-Path framework has only limited ability of load-balancing the traffic). Therefore, multipath routing has been introduced to a limited extent in operational networks.

The emergence of new concepts for the next generation Internet is believed to offer significant opportunities for the application of multipath routing. This refers in particular to *content-centric networks* (CCN) [6] due to the implied one-to-any communication model. In this paper we study the viability of joint use of load-balancing, multipath routing and caching in this architecture with the emphasis on decentralized solutions.

There are a few sidelights that multipath routing can improve the performance of CCN when compared to single-path routing. In particular, one can expect that even if multipath routing results in long paths, data can potentially be cached somewhere along the path towards the origin server. As a result, using multipath routing in CCN networks does not have to mean that on average data is transmitted on long paths, at least as far as it concerns popular data, i.e. one having a high probability to be cached by content routers. Moreover, the number of available paths/routes between a given access node and the origin server influences the probability of requested data to be sent from network cache, which offers some level of control of network performance. For instance, one could forward all requests for same data item to a single next hop. In that case, the popularity of particular data item would typically increase along the path towards the origin server and potentially improve caching efficiency. Conversely, requests for unpopular content can be spread over multiple paths in order to additionally decrease its popularity and limit the pollution of the caches along those paths.

We note however that the overall network effect of using such balancing strategies in CCN architecture has not been subject to systematic studies. In particular, in [7] no algorithmic approaches are provided. A first comparative study of several caching policies under multipath routing has been presented in [8], but the authors do not consider adaptive load balancing so their conclusions are not fully applicable to our problem. Other previous works that combine caching and load balancing refer either to managed CDNs [9] and result in centralized solutions, or to some regular network structures [10] that are derived from computer architectures and are not general enough for networks like the Internet.

The contribution of this paper is two-fold. First, we show that multipath routing and link-level load balancing can be effectively combined with caching into a joint mechanism that helps decrease network delays. This framework is particularly suited to flash crowd conditions. Second, we propose quite general popularity-based load balancing mechanism that makes use of available multiple paths, is local in nature and is as scalable as other popularity-based strategies, e.g., LFU.

The paper is organized as follows. In section II, the model of content network is presented. The concept of combined multipath routing and load balancing in the content networks is described in Section III, and the results of simulation experiments are presented in Section IV. The paper is concluded in Section V.

II. Network model

In this paper we consider the original CCN framework [6] as a reference model of a content-centric network. We define three types of nodes $\mathcal{V}$: access nodes $\mathcal{V}_a \subset \mathcal{V}$ that host the users, content routers $\mathcal{V}_c \subset \mathcal{V}$, and origin servers $\mathcal{V}_o \subset \mathcal{V}$ being the sources of data. The nodes are connected by directed links $\mathcal{E}$. Subsets $\mathcal{E}_+(v) \subset \mathcal{E}$ and $\mathcal{E}_-(v) \subset \mathcal{E}$ contain links incoming to and outgoing from node $v \in \mathcal{V}$, respectively.

Following [6] we assume that content delivery is based on chunk transmissions, i.e., each chunk has its own name and from the network perspective it constitutes a separate object. In the following, the set of all chunks is denoted by $\mathcal{C}$.

Upon the reception of an interest (i.e., a request for a chunk), content router checks if the corresponding chunk is available in the local cache and if so it immediately sends back the requested chunk. Otherwise, content router registers the interest in PIT (Pending Interest Table) and passes it to next hop nodes according to the entries available in the forwarding table and the ultimate decision of the strategy layer (the latter being responsible for making actual decisions as to the forwarding of interests). Upon the reception of requested data from any incoming interface, the content router passes it to all outgoing interfaces pointed by the corresponding entry of PIT (deleted afterwards) and updates its cache if necessary. Since CCN concept is routing agnostic, using multipath routing is an option that depends on the strategy layer.

Soft state PIT provides a simple mechanism that prevents content routers from replicating already seen interests, and also helps in eliminating loops in routing paths. As looping paths may degrade network performance, we consider only loop free routing patterns. More specifically, we use single shortest paths as reference routing patterns, and *equal cost multipath* (ECMP) [11] for determining multipath routing. For a given data item $c \in \mathcal{C}$, related forwarding table in content router $v \in \mathcal{V}_c$ is defined by a subset $\mathcal{R}(v;c) \subset \mathcal{E}(v)$ of node v interfaces that can be used to pass interests for this data item c to next hops. We assume that all paths determined by $\mathcal{R}(v;c)$ are equivalent to each other from the perspective of the routing protocol, and as has already been mentioned, it will be the role of CCN strategy layer to define the right use of those paths.

CCN framework does not make specific assumptions as to which caching policy is to be used by content routers, and consequently any caching policy should be applicable. As our main goal is to investigate how using proposed variants of multipath routing can influence performance of the content network, we focus on two replacement policies: LRU (often considered in CCN context for its simplicity) and PBLRU (Popularity Based LRU).

LRU is relatively easy to implement as it does not require explicit popularity detection. However, if the number of unpopular data items (in extreme case one-timers) is large, the efficiency of such replacement policy can be questionable. PBLRU is a modification of LRU that is able to exploit known estimates of content popularity. More specifically, under PBLRU the so-called one timers and unpopular data are prevented from being passed to proper policy—LRU based on initial filtering. Certainly, explicit popularity estimation introduces computational overhead compared to simple LRU, but our numerical experiments suggest that the number of data items that need such estimation can be quite small in case of PBLRU. So, we believe that PBLRU would not be an implementation challenge in real systems and skip a more detailed discussion on this topic.

III. Load balancing

One of the general principles of using multipath routing locally in a content router is balancing the load due to data items on the incoming interfaces. CCN architecture brings one feature that is particularly important for this purpose. Namely, forwarding an interest at specific outgoing interface implies that the corresponding data will arrive at the corresponding incoming interface. So, managing proportions of interests sent at different outgoing interfaces is a mechanism to control the load on corresponding incoming interfaces. This observation is an essence of the mechanisms proposed in the following.

Having given the load of incoming interfaces $\boldsymbol{l} = \{l_e : e \in \mathcal{E}_+(v),\ v \in \mathcal{V}_c\}$, a content router can balance their load by appropriately spreading the interests that it forwards through corresponding outgoing interfaces. Then an intuitive choice is to forward an interest to the least loaded interface, defined as:

$$e = \arg\{\min_{e \in \mathcal{E}_+(v),\, v \in \mathcal{V}_c\}} l_e\}. \qquad (1)$$

We refer to the above scheme as *minimum load first* (MLF). Although MLF tends to balance the load of incoming interfaces, the requests for same data item may traverse different paths in longer term. So, using MLF can decrease the probability that requested data is cached in the next hop node.

In order to avoid the negative effect as indicated above and keep caching probability high, we consider another scheme referred to as *modulo hash first* (MHF). Similarly to ECMP [2], MHF calculates a hash $H(c)$ of data identifier and determines the outgoing interface by applying modulo operation upon obtained hash value. Thus, all interests requesting specific data item are always sent through the same outgoing interface. Consequently, the paths used to send those interests merge and create a tree structure. Although it may be considered as constraint, using tree routing allows to aggregate interests for a given data object and maximize its cacheability. In this case (and assuming $\mathcal{R}(v,c)$ to be an ordered list), the selection of an outgoing interface can be expressed as:

$$e = \mathcal{R}(v,c)[idx] \text{ where } idx = H(c) \bmod |\mathcal{R}(v,c)| \qquad (2)$$

MHF uses a unique tree to transmit all interests related to a specific data identifier thus trying to maximize the probability that this item is cached in the network. The accuracy of resulting load balancing depends on the granularity of traffic flows associated with particular modulo groups. We suggest that the precision of resulting load balancing can be improved by applying MLF to least popular content - we use this idea in the following to define PALB strategy.

As has already been mentioned, our approach is based on data popularity estimation. Proposed scheme is referred to as *popularity aware load balancing* (PALB), an is assumed to partition data items into three groups depending on data popularity. PALB tries to merge the benefits of both MLF and MHF in order to maximize the effectiveness of caching while balancing link load.

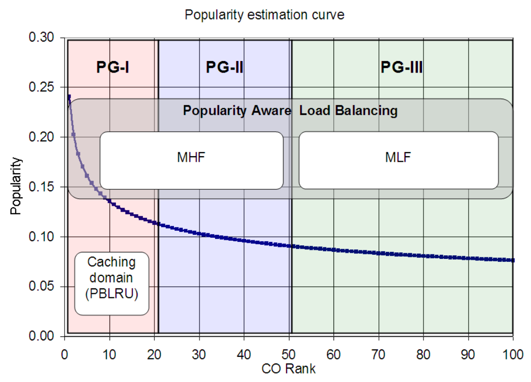

Fig. 1: PALB content partitioning

Upon the reception of a message (interest or data) related to particular content, content router using PALB associates it with one of three popularity groups: high-, medium- and low-popularity group (PG-I, PG-II, PG-III) (such partitioning is illustrated for a sample Zipf distribution in Fig. 1). Interests are served according to algorithm III.1, and if a request cannot be served from the local cache it is forwarded depending on its popularity group association as follows: according to MHF rule if the group is PG-I or PG-II, or according to MLF rule if the group is PG-III. For data messages, only those belonging to PG-I are directed to LRU cache if PBLRU caching is used (in this way only the most popular content is input to internal LRU cache - see algorithm III.2).

Algorithm III.1: PROCESSCONTENTREQUESTPALB(c)

```
UPDATEPOPULARITYSTATS($c$)
if CONTENTINCACHE($c$) = true
  then { SENDDATABACKTOINCOMMINGFACEFROMCACHE($c$)
         comment: c on top of LRU cache
  else { if REQUESTINPIT($c$) = false
           then { REGISTERINPIT($c$)
                  if PG($c$) = $I$ or PG($c$) = $II$
                    then FWDREQUESTTOLINK(MHF(H($c$)))
                    else FWDREQUESTTOLINK(MLF())
           else REGISTERINCOMMINGFACEINPIT($c$)
```

Algorithm III.2: PROCESSCONTENTDATA(c)

```
if $CachingType = PBLRU$ and PG($c$) = $I$
  then STOREDATAINCACHE($c$)
if $CachingType = LRU$
  then STOREDATAINCACHE($c$)
FWDDATABACKTOALLINCOMMINGFACESINPIT($c$)
```

As a general rule, the use of high popularity (PG-I) items works in favour of maximizing the efficiency of caching. In this work, we consider PBLRU to be an efficient add-on to LRU (assuming that popularity estimates are already available from PALB), but using other policies is also possible. Moreover, PG-II content is moderately popular, and PG-II interests should preferably be transmitted through deterministic paths to increase their caching probability along the paths. Finally, PG-III interests (associated with unpopular and one-timer data items) will typically not benefit from caching and should be prevented from caching if possible; spreading such interests among many paths is a means to achieve this goal. PG-III content is thus perfectly suited to realize local principles of load balancing. Based on data popularity estimation, we can achieve good caching efficiency by preserving high probability of caching for high/medium popularity traffic, and accurate load balancing with the use of uncachable PG-III traffic.

Proposed load balancing mechanism is controlled by data popularity. Although in our experiments we assume that data popularity obeys Zipf's law, our framework does not preclude other distributions. More information on popularity detection with regard to our framework is contained in section IV-C.

IV. SIMULATION EXPERIMENTS

In order to reflect the operation of CCN network we have created a dedicated simulator. In the conducted numerical experiments we have paid particular attention to reconstruct traffic scenarios featuring the presence of hot spots in the network.

A. Network models used in simulation experiments

The simulation experiments exploit two network topologies: *Diamond* and *Mesh*, both depicted in Fig. 2. The Diamond topology consists of seven content routers in a specific layout so that the load balancing decisions are taken only in one content router CR0. We consider one access nodes and one origin server connected by five possible paths and a single origin-destination traffic demand. This simple case is used to gain basis intuition regarding the performance of multipath routing schemes defined in this paper. More realistic traffic

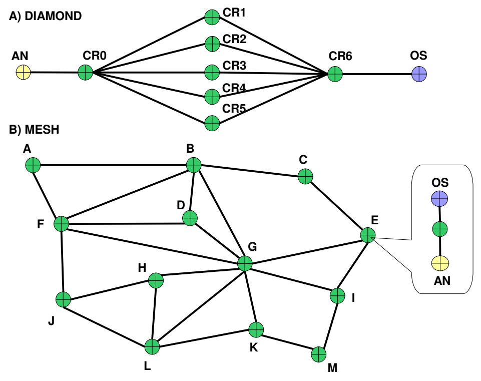

Fig. 2: A) Diamond and B) Mesh network topologies

scenarios, containing sink- and source- hot spots in particular, are investigated for more complex Mesh topology of the network. In the Mesh topology, each content router works in the caching mode, and to each of them, one access node (sink of traffic) and one origin server (source of traffic) are connected.

Capacity of links connecting a pair of content routers is equal to 100Mbps, and capacity of links connecting content routers and access/origin nodes is equal to 1000Mbps. Such a selection of capacities is deliberate in order to eliminate bottlenecks between and content routers and allow to study only the performance on network level.

We assume that the size of link buffers is limited so that the maximum queuing delay on a single link is $ld = 40$ms. This assumption is in line with known practices in networking, where line card buffers are relatively small in order to have short delays and low jitter (bufferless multiplexing) The maximum queuing delay directly influences the time after which interests are cleared from PITs in content routers when corresponding data messages do not arrive (because of loss). For the Mesh topology, one can easily see that the longest ECMP-compliant path between any two content routers consists of 4 hops. So, given the worst case (interest message and data message are maximally delayed by each link on the longest path), the source router should wait at most 320ms ($8 * 40$ms) for the data packet. Taking into account processing time in each cache router as well as transmission time, and propagation delays, we (somewhat arbitrarily) doubled 320ms and set the Time To Clear (TTC_c) for interest messages in PITs to 640ms.

In our simulator, each data request (to get a chunk) is issued by an application process. We assume a conservative congestion control strategy at this level when transmitting whole data objects: the application process issues an interest for a chunk only after the reception of the previous chunk. An interest for a particular chunk is renewed if the corresponding data does not arrive within 800ms. This value stems from adding to TTC_c the maximal delay on the access links to content routers (800ms = 640ms + $4 * 40$ms).

B. *Traffic model*

We consider a set of $|\mathcal{C}| = 1000$ data items available across the network, each of size $s_c = 1$MB. We adopt Zipf's distribution presented in [12] and [13] for modelling global data popularity distribution $\boldsymbol{\gamma}$ with various values of Zipf skew parameter $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. The set $\mathcal{C}$ is ordered according to decreasing popularity, and γ_c represents the probability the data item of rank $c = \{1, ..., 1000\}$ is requested. Clearly, 1000 data items is not representative for the current Internet, but after having initially experimented with a much larger set of objects we found that 1000 objects and the resulting granularity of traffic flows is sufficient to quantitatively compare investigated mechanisms while not overburdening the simulator.The equal size of data items was assumed in order to sharply expose the influence on network performance of pure multipath/load balancing mechanisms.

The discussion on the selection of suitable chunk size in content oriented networks with built-in caching is ongoing. Different architectures have their own assumptions on that. Following several works related to CCN [14], [15], we have assumed 10kB as the chunk size.

Poisson distribution is used to model request arrival process $\boldsymbol{\lambda}$ in access nodes, with parameter λ^G for the whole network. We assume that each content router can cache up to 3% of the total number of data items which translates into 30MB of caching memory.

For Diamond topology, we assume that all data items are requested by one access node and retrieved from one origin server. For Mesh topology we model source-hot spots scenarios by associating origin servers with particular data items. In our experiments, we assume that each origin server $v \in \mathcal{V}_o$ hosts equal number of data items ($k = \frac{|C|}{|\mathcal{V}_o|} = 1000/13 = 76$). We also assume that the set of origin servers ($\mathcal{V}_o$) is ordered, and the first origin server on the list contains the first k data items from the set $\mathcal{C}$, the second origin server contains the next k data items, and so on. Thus, as the set $\mathcal{C}$ is ordered according to decreasing data popularity, the source-hot spot(s) are located in the first origin server(s) in the set $\mathcal{V}_o$. We must notice that for different Zipf skew parameters of popularity distribution, we obtain different distributions of requests directed to particular origin servers $\gamma_o = \sum_{c \in \mathcal{C}_o} \gamma_c$, where $\mathcal{C}_o$ denotes a subset of data items associated with origin server o (see Table IA). For many simulation runs, we randomize the order in the set $\mathcal{V}_o$, thus modelling different locations of data items in the network topology (and thus the locations of source-hot spots).

TABLE I: Source-hotspot (A) and sink-hotspot scenarios (B).

A) Source hot-spot scenarios - γ_o distribution.													
Zipf Skew \ OS Idx	1	2	3	4	5	6	7	8	9	10	11	12	13
0.2	0.127	0.095	0.086	0.080	0.076	0.073	0.071	0.069	0.067	0.066	0.064	0.063	0.061
0.4	0.207	0.112	0.090	0.079	0.071	0.066	0.062	0.058	0.055	0.053	0.051	0.049	0.047
0.6	0.326	0.120	0.087	0.071	0.061	0.054	0.049	0.045	0.042	0.039	0.037	0.035	0.033
0.8	0.485	0.114	0.075	0.057	0.046	0.039	0.034	0.031	0.028	0.025	0.023	0.022	0.020
1	0.658	0.092	0.054	0.038	0.030	0.024	0.021	0.018	0.016	0.014	0.013	0.012	0.011
B) Sink hot-spot scenarios - σ_a distribution.													
AN Idx	1	2	3	4	5	6	7	8	9	10	11	12	13
Scenario A	0.143	0.132	0.121	0.110	0.099	0.088	0.077	0.066	0.055	0.044	0.033	0.022	0.011
Scenario B	0.830	0.146	0.022	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Sink-hot spot scenarios are modelled by adjusting request intensity proportions between different access nodes ($\boldsymbol{\sigma}$ distribution). In order to generate various simulation conditions, we randomize the order of the set $\mathcal{V}_a$ and use the function $f(i_v) = xi_v^y + z$, where i_v is the index of node v in set $\mathcal{V}_a$, and x, y, z are tunable parameters. Thus, $\sigma_v = \frac{f(i_v)}{\sum_{v \in \mathcal{V}_a} f(i_v)}$. Scenario A (Table IA), defined by parameters $x = 1, y = 1, z = 1$, models moderately differentiated (within an order of magnitude) proportions of request rate in particular access nodes. Scenario B (Table IB), defined by parameters $x = 1, y = 20, z = 1$, in practice limits the number of active access nodes in the network to two sink-hot spots, where the first node generates 83% and the second nearly 15% of the total traffic in the network. So, we may say that Scenario B is an extreme case of sink-hot spot.

Each simulation run lasts 2000s with 200s warm up period. In case of Diamond topology it corresponds to 20-30 thousands of content requests, and 2-3 millions of satisfied interests for respective chunks (1 data item consists of 100 chunks). In case of Mesh topology it corresponds to 50 and 240 thousands of content requests (sink hot-spot scenario B and A, respectively), and 5-24 millions of satisfied interests for respective chunks. For Mesh topology, we simulated 10 random configurations in terms of source and sink-hot spots, so the presented values are respective averages.

Referring to the elementary mechanisms defined in the previous section, by a network configuration scheme we understand a network that uses particular type of caching policy (LRU, PBLRU), routing (SP - single shortest path, MP - equal cost multipath) and load balancing solutions (MLF, MHF, PALB(p))[1] in case of multipath routing. The convention we adopt for naming the schemes is of the form `Caching_Routing_LoadBalancing`. For instance, `LRU_MP_MHF` denotes the network scheme that uses LRU for caching, ECMP multipath routing and modulo MHF load balancer.

Average file retrieval time (retrieval time (RT) for short) is the main criterion we use to compare the efficiency of various network configuration schemes. It is expected to be a good synthetic performance indicator, because it is influenced by both caching and transport efficiencies. Another criterion, called *cache hit rate* (CHR), is used as a secondary performance descriptor. It represents the percentage of interests served by network caches (so excluding the origin servers). The last criterion, called *network balance indicator*, is defined as follows. For a given simulation run (and a given traffic matrix) we calculate link load peakedness factor as the ratio of the variance to the mean of link loads in the network. Then network balance indicator is obtained by averaging link load peakedness factor over 10 simulation runs, each time assuming a distinct traffic matrix being a random permutation of sink-hot spot Scenario B (c.f. Table IB).

C. *Popularity estimation and PG-I size selection*

We recognize that the accuracy of popularity estimation is of crucial importance for the performance of PALB. As tracking the popularity of all data is impossible in wide deployments, pragmatic methods may use finite data structures, like fixed-length lists, to count the frequencies of requests related to a subset of data items. In order to provide a working method for our experiments, below we propose a simple estimation of PG-I and PG-II based on observing a relatively small number of data items.

In our experiments, each content router maintains popularity list of data items and registers the number of requests for each data item c from the list in each consecutive measurement window k; this number is denoted as (denoted by r_c^k). Popularity estimate pe_c^k during measurement window k for data item c is then calculated using exponential moving average according to $pe_c^k = pe_c^{k-1}(1-\alpha) + \alpha r_c^k$. We tuned the values of the measurement window duration experimentally and the smoothing factor α to 1s and $\alpha = 0.05$, respectively.

In order to define popularity groups PG-I and PG-II we assume that at any moment they are given together as fraction p of the most popular data items that are actually monitored in popularity list. Accordingly, by PALB(p) we denote a scheme where fraction p of the most popular data (currently in popularity list) belong to PG-I and PG-II (we remind that the requests for their chunks are load-balanced according to MHF). Remaining data items constitute the low popularity set PG-III and are balanced according to MLF. The size of PG-I refers only to PBLRU and is commented separately.

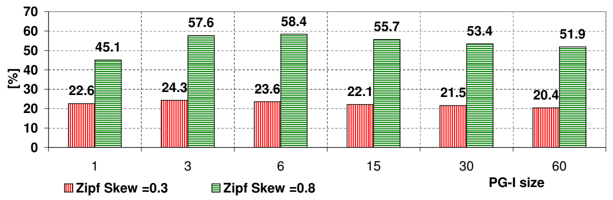

Fig. 3: Cache hit rate (CHR) as a function of PG-I size. (Diamond topology, PBLRU_MP_MHF, Zipf skew = 0.3 and 0.8, $\lambda^G = 10$).

Popularity-driven policy PBLRU is an enhancement to LRU scheme assuming that only chunks of data belonging to popularity group PG-I are fed to proper LRU cache. The efficiency of PBLRU scheme thus depends on PG-I size. After many preliminary experiments under different assumptions concerning popularity measurement, the size of PG-I that led to the maximal hit ratio CHR was within the range 3–10 data items for both considered topologies. Exemplary results of such experiments for Diamond topology are presented in Fig. 3. We can see that the optimal size of PG-I is 3 or 6 depending on assumed Zipf skew parameter of popularity distribution (0.3 and 0.8 respectively). We explain those results as follows: for small sizes of PG-I list, the caches tend to be underutilized (not full) which results in a drop of cache hit ratio; under a very large size of PG-I list, PBLFU works like a classical LRU, because majority of chunks are fed directly to the cache and increase cache pollution with unpopular items (which also leads to a drop of the hit ratio). It has to be explained that PG-I consists of data items with the highest value of estimated popularity in consecutive measurement windows. Due to the random process of request arrival and temporal variability of popularity estimations, PG-I list is also variable and captures a wide set of relatively popular data items in longer run. This set is fed to the LRU cache. Thus, even if the size of PG-I is relatively small, the LRU cache is filled completely if PG-I size is tuned appropriately. For all simulation runs presented in the next sections we assumed constant size of PG-I list equal to 5.

We realize that popularity detection is in general a complex and difficult to scale task. Therefore, the above suggestion that the size of PG-I can be relatively small is promising and allows to formulate a working hypothesis that the number of data items for which popularity estimation is performed could also be small in reality. We leave a detailed investigation of this topic for future work.

[1]the notation PALB(p) is explained in the following subsection IV-C

D. Results for Diamond topology

In Fig. 4 and Fig. 5, plots for RT and CHR related to the Diamond topology for various network schemes using LRU caching policy and $\lambda^G = 10$ are drawn. Poor performance of `LRU_SP` results from the fact that the whole traffic is transmitted using single path (AN-CR0-CR1-CR6-OS), and cache stores in content routers CR2-5 are not used at all. This is evident from Fig. 6 which shows that for scenario `LRU_SP` only CR0, CR1, CR6 participate in data forwarding.

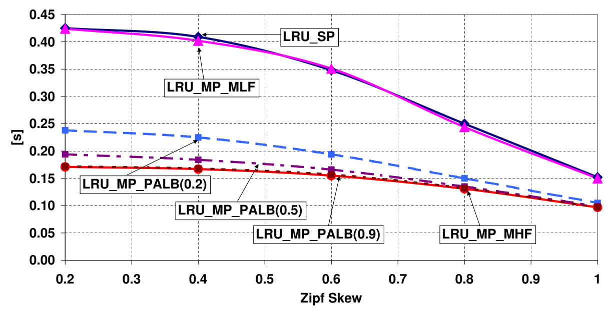

Fig. 4: Average file retrieval time (RT) as a function of Zipf skew for LRU (Diamond topology, $\lambda^G = 10$).

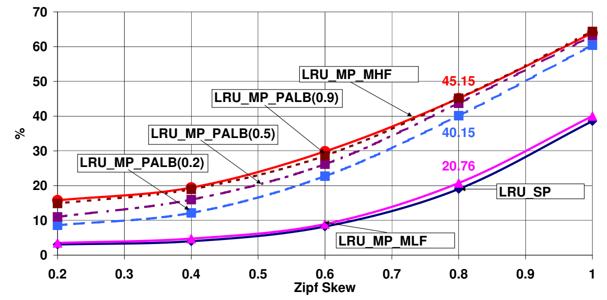

Fig. 5: Cache hit rate (CHR) as a function of Zipf skew for LRU (Diamond topology, $\lambda^G = 10$).

Although `LRU_MP_MLF` scheme uses multipath routing its efficiency is similar to `LRU_SP`. This is so because MLF, due to load balancing, forwards successive requests for a given chunk onto different paths. This lowers the probability that the next request for this chunk is directed to content router that has stored it previously. In effect, performance of caching in content routers CR(1-5) deteriorates. In Fig. 6 we can see that all CRs detected requests for each data item. Thus, multipath schemes with MLF load balancing play against caching and offer poor performance in terms of RT and CHR.

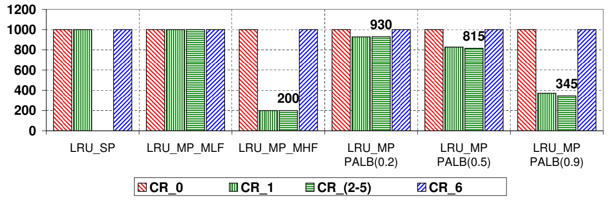

Fig. 6: Detected number of content items in particular CRs (Diamond topology, $\lambda^G = 10$, Zipf Skew=0.8).

We observe that the most efficient network scheme is `LRU_MP_MHF`. We must remember that use of MHF physically separates interests among modulo groups seen by content routers CR(1-5). Thus, the number of data items seen locally in each content router CR(1-5) is limited to 200 (c.f. Fig. 6). In that case, we can get high hit rates in network caches due to low rates of pre-emption and low RT.

Schemes `LRU_MP_PALB(p)` ($p \in \{0.2, 0.5, 0.9\}$) preserve modulo paths (MHF) for the most ranked content, but spread (MLF) interests for unpopular content. Clearly, `LRU_MP_PALB(p=0)` corresponds to `LRU_MP_MLF`, and `LRU_MP_PALB(p=1)` corresponds to `LRU_MP_MHF`. In fact, the higher the value of parameter p (the size of PG-I and PG-II) is, the lower the number of data items as seen locally at each content router CR(1-5) (c.f. Fig. 6, 930 for $p = 0.2$, 815 for $p = 0.5$, 345 for $p = 0.9$) and the better caching efficiency due to the low eviction rates.

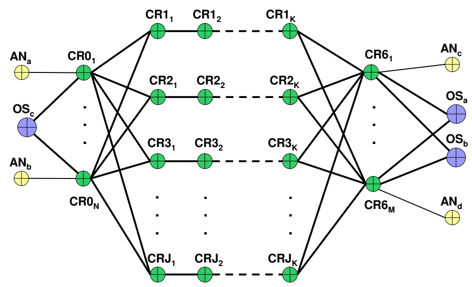

Fig. 7: Extended Diamond topology appropriate for the use of MHF load balancing.

Taking into account the computational overhead (for popularity detection) required by `LRU_MP_PALB(p)` schemes, the use of such schemes for topologies having similar properties that of Diamond topology (see Fig. 7) is unprofitable comparing to simple `LRU_MP_MHF`. However, in more complex topologies with multi-homed access nodes and origin servers (like in case of Mesh topology), the aggregation by limiting the number of seen data items to improve caching efficiency is relatively harder. In such a case, we can still expect a positive impact of the use of popularity aware scheme PALB described in Section IV-E).

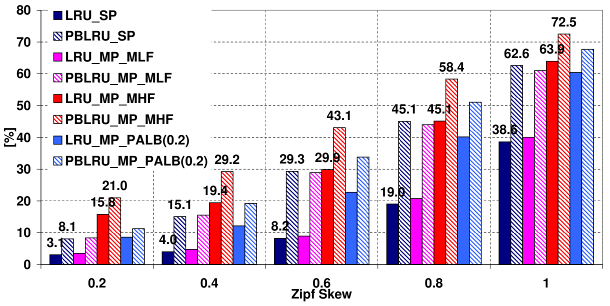

Fig. 8: Cache hit rate (CHR) as a function of Zipf skew (Diamond topology, $\lambda^G = 10$).

In Fig. 8 and Fig. 9, we can observe how replacing LRU by PBLRU influences CHR and RT. Using PBLRU significantly improves CHR (Fig. 8), and thus RT can be much lower (Fig. 9). This is particularly visible for network schemes `LRU_MP_MLF` and `LRU_SP`, for which RT was reduced from $0.25s$ to $0.18s$ (Zipf skew$=0.8$). Comparing the best scenarios, i.e., `PBLRU_MP_MHF` and `LRU_MP_MHF`, we infer that quite high increase of CHR (from 45.1 to 58.4, Zipf skew= 0.8, Fig. 8) results in moderate decrease of RT (from 0.13 to 0.11, Zipf skew= 0.8, Fig. 9). The reason is that CHR is an aggregated indicator that reflects the percentage of interests served by any network cache, but without information on how close to the access node (in terms of hop counts) the ache was located. It is obvious that RT is shorter if hits happen in nearby caches. Therefore, RT is superior to CHR as a synthetic indicator of efficiency in a network of load-balanced caches.

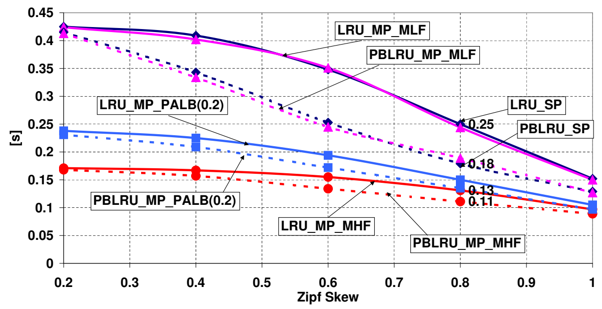

Fig. 9: Average file retrieval time (RT) as a function of Zipf skew (Diamond topology, $\lambda^G = 10$).

E. Results for Mesh topology

In Fig. 10 we compare retrieval time RT in the Mesh topology for various network schemes and sink-hot spot Scenario A (c.f. Table IA). Obviously, schemes: `LRU_SP` and `LRU_MP_MLF` are much less efficient than `LRU_MP_MHF` and `LRU_MP_PALB(0.2)`.

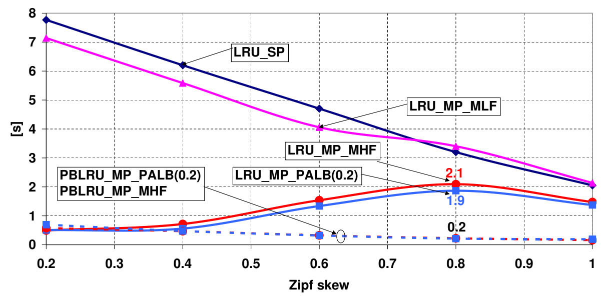

Fig. 10: Average file retrieval time (RT) as a function of Zipf skew (LRU / PBLRU caching, Sink-hot spot Scenario A, $\lambda^G = 120$).

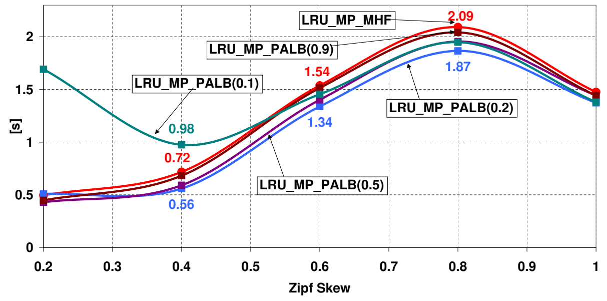

Fig. 11: Average file retrieval time (RT) as a function of Zipf skew (LRU caching, Sink-hot spot Scenario A, $\lambda^G = 120$).

However, we can see that `LRU_MP_PALB(0.2)` slightly outperforms `LRU_MP_MHF` (now, RT equals $1.9s$ as compared to $2.1s$, Zipf skew=0.8). As expected, such popularity-aware load balancing shapes the popularity seen in next hop content routers so that their high caching efficiency is maintained.

The above effect is investigated in details for LRU schemes in Fig. 11. As can be seen, there exists an optimal value of parameter p (see Section III) characterizing PALB, for which we observe the lowest value of RT (`LRU_MP_PALB(0.2)`). Setting the value of p too small, as for `LRU_MP_PALB(0.1)`, degrades the efficiency for lower values of Zipf skew.

The application of PBLRU (instead of pure LRU) gives additional gain in the form of reduced RT (c.f. Fig. 10). Both schemes `PBLRU_MP_PALB(0.2)` and `PBLRU_MP_MHF` significantly decrease RT as compared to their counterparts based on LRU (RT=$0.2s$ vs. $1.9s$ and $2.1s$, Zipf skew=0.8), however, there are no significant differences between them.

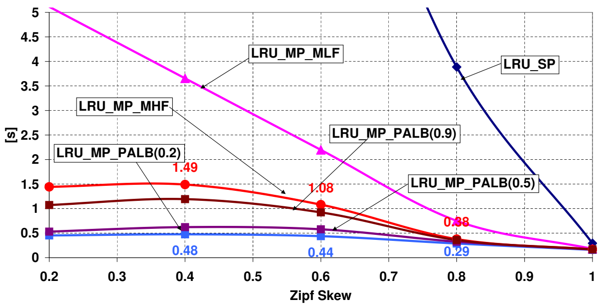

Fig. 12: Average file retrieval time (RT) as a function of Zipf skew (LRU caching, Sink-hotspot Scenario B, $\lambda^G = 25$).

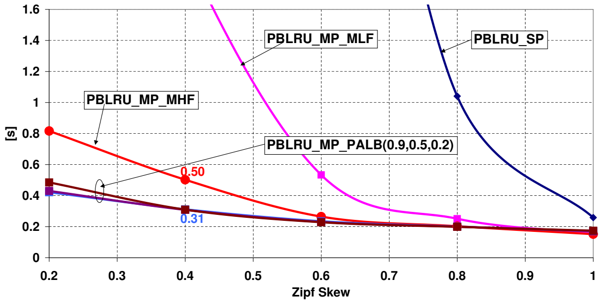

Fig. 13: Average file retrieval time (RT) as a function of Zipf skew (PBLRU caching, Sink-hotspot Scenario B, $\lambda^G = 25$).

A true benefit of using PALB as compared to MLF is evident in Fig. 12 (LRU caching) and Fig. 13 (PBLRU caching) which depict the RT values for imbalanced sink-hot spot Scenario B (c.f. Table IB). Comparing `LRU_MP_PALB(p)` (especially for $p = 0.2$) and `LRU_MP_MHF` we observe (for lower values of the Zipf skew parameter) that RT can be decreased from $1.5s$ to $0.5s$ and from $0.4s$ to $0.3s$ when Zipf skew parameter is equal to 0.4 and 0.8, respectively. Thus, a significant reduction of RT (c.f. Fig. 12) is achievable.

Comparing schemes `PBLRU_MP_PALB(p)` and `PBLRU_MP_MHF` for PBLRU caching we can still observe a positive impact of using PALB, but this holds mainly for lower values of Zipf skew parameter (c.f. Fig. 13). RT can be decreased from $0.5s$ to $0.3s$ due to PALB when Zipf skew parameter is equal to 0.4, but for higher values of Zipf skew parameter the gain is not significant. We thus conclude that aggregation helps mainly in scenarios with moderately skewed popularity distributions. However, replacing LRU by PBLRU improves caching efficiency and thus reduces RT for any load balancing scheme.

Finally, we check network balance indicator in order to confirm the desired effect of joint operation of routing, load balancing and caching. To this end, in Fig. 14, network balance indicator is drawn in function of Zipf skew for various load balancing schemes assuming PBLRU caching. It can be observed that PALB outperforms the remaining schemes, especially the one based on single shortest path routing.

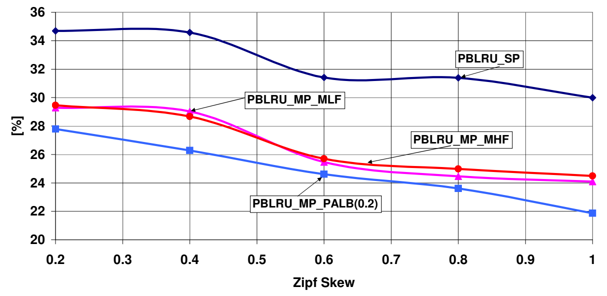

Fig. 14: Network balance indicator as a function of Zipf skew (PBLRU caching, Sink-hot spot Scenario B, $\lambda^G = 25$).

V. Conclusions and future work

Applying load balancing in content networks affects efficiency of content caching. Popularity aware load balancing can be thus a way to optimize caching efficiency. Caching efficiency depends on skewness of content popularity characteristics. In general, higher skewness of characteristics implies higher caching efficiency. Observable by network nodes popularity of content items can be aggregated by using tree-like routing structures to transmit requests and data. Thus, on one hand, load balancing when applied to popular content should preserve aggregated popularity characteristics, e.g., through using tree-like routing structures and balance load on content level, i.e., in ECMP fashion use unique paths to all requests related to particular content item. On the other hand, popularity of unpopular content (noise from caching efficiency perspective) can be further decreased by using random paths to transmit content items independently. Results of simulations presented in the paper show that using proposed approach can have positive impact on network performance measured in terms of file retrieval time, in particular for hot-spotted traffic scenarios.

The efficiency of caching methods typically depends on the number of observable content items. When popularity estimates are already available, the efficiency of caching can be increased significantly by filtering out unpopular content. The latter possibility is successfully exploited by a straightforward popularity-aware extension of LRU proposed in the paper.

Estimation of content popularity is a challenging task. For the sake of performed simulations, simple popularity estimation method was proposed. In the proposed approach popularity was estimated as a moving average of time-framed request counters. We may conclude that this approach requires adjusting measurement interval to properly smooth popularity characteristic in required time scale. From the perspective of the load balancing approach proposed in the paper, one remark seems to be important. Namely, the objective is not to measure popularity of particular content, but to determine popularity level (popularity group) requested content belongs to, and to apply appropriate load balancing rule. Although popularity estimation is not easy, taking into account potential benefits, we should not treat this issue as impossible and find it as important topic for the future work.

References

[1] R. Banner and A. Orda, “Multipath routing algorithms for congestion minimization,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 413–424, Apr. 2007.

[2] M. Dzida, M. Zagozdzon, M. Pioro, and A. Tomaszewski, “Optimization of the Shortest-Path routing with Equal-Cost Multi-Path load balancing,” in *2006 International Conference on Transparent Optical Networks*, vol. 3. IEEE, Jun. 2006, pp. 9–12.

[3] I. Cidon, R. Rom, and Y. Shavitt, “Analysis of multi-path routing,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 885–896, Dec. 1999.

[4] E. Kubilinskas, F. Aslam, M. Dzida, and M. Pioro, “Recovery, routing and load balancing strategy for an ip/mpls network,” *Lecture Notes in Computer Science*, vol. 4516, 2007.

[5] A. Tomaszwski, M. Pioro, M. Dzida, M. Mycek, and M. Zagozdzon, “Valid inequalities for a shortest-path routing optimization problem,” in *Proceedings of International Network Optimization Conference (INOC)*, Spa, Belgium, 2007.

[6] 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*. Rome, Italy: ACM, 2009, pp. 1–12. [Online]. Available: http://portal.acm.org/citation.cfm?id=1658941

[7] S. Dibenedetto, C. Papadopoulos, and D. Massey, “Routing policies in named data networking,” in *Proc. ACM SIGCOMM ICN'11*, Toronto, Ontario, Canada, 2011.

[8] D. Rossi and G. Rossini, “Caching performance of content centric networks under multi-path routing (and more),” Telecom ParisTech, Tech. Rep., 2011.

[9] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, “Optimal content placement for a large-scale vod system,” in *Proceedings of the 6th International Conference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 4:1–4:12.

[10] F. Meyer auf der Heide, B. Vöcking, and M. Westermann, “Caching in networks (extended abstract),” in *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '00. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000, pp. 430–439. [Online]. Available: http://dl.acm.org/citation.cfm?id=338219.338589

[11] C. B. Garcia-Luna-Aceves, C. Balasubramanian, and J. J. G. luna aceves, “Shortest multipath routing using labeled distances,” in *In Proceedings of the IEEE International Conference on Mobile Ad hoc and Sensor Systems*, 2004.

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distributions: Evidence and implications,” *IN INFOCOM*, pp. 126—134, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.8742

[13] M. Jauhari, A. Saxena, and J. N. Gautam, “Zipf's law and number of hits on the world wide web,” *Annals of Library and Information Studies*, vol. 54, no. 2, pp. 81–84, 2007. [Online]. Available: http://cat.inist.fr/?aModele=afficheN&cpsidt=18979608

[14] L. Muscariello, G. Carofiglio, and M. Gallo, “Bandwidth and storage sharing performance in information centric networking,” in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 26–31. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018593

[15] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, “Modeling data transfer in content-centric networking,” in *Teletraffic Congress (ITC), 2011 23rd International*. IEEE, Sep. 2011, pp. 111–118.