Journal of Network and Systems Management (2020) 28:298–339
https://doi.org/10.1007/s10922-019-09507-x

Load Balancing in the Presence of Services in Named-Data Networking

Dima Mansour[1] · **Haidar Osman**[2] · **Christian Tschudin**[1]

Received: 3 April 2019 / Revised: 6 December 2019 / Accepted: 11 December 2019 /
Published online: 21 December 2019

Abstract
Load balancing is a mechanism to distribute client requests among several service instances. It enables resource utilization, lowers response time, and increases user satisfaction. In Named-Data Networking (NDN) and NDN-like architectures, load balancing becomes crucial when dynamic services are present, where relying solely on forwarding strategies can overload certain service instances while others are underutilized especially with the limited benefit of on-path caching when it comes to services. To understand the challenges and opportunities of load balancing in NDN, we analyze conventional load balancing in IP networks, and three closely related fields in NDN: congestion control, forwarding strategies, and data center management. We identify three possible scenarios for load balancing in NDN: facade load balancer, controller for Interest queues, and router-based load balancing. These different solutions use different metrics to identify the load on replicas, have different compliance levels with NDN, and place the load balancing functionality in different network components. From our findings, we propose and implement a new lightweight router-based load balancing approach called the communicating vessels and experimentally show how it reduces service response time and senses server capabilities without probing.

Keywords Service-Centric Networking (SCN) · Information Centric Networking (ICN) · Forwarding strategies · Congestion control

✉ Dima Mansour
dima.mansour@unibas.ch

Haidar Osman
haidar.osman1@swisscom.com

Christian Tschudin
christian.tschudin@unibas.ch

[1] Department of Mathematics and Computer Science, University of Basel, Spiegelgasse 1, 4051 Basel, Switzerland

[2] Swisscom AG, Waldegstrasse 51, Liebefeld, 3097 Bern, Switzerland

1 Introduction

IP addresses are the building blocks of today's Internet architecture. Published content and services are reached by establishing end-to-end connections between clients and servers. This architecture is questioned [1, 2] since many research studies illustrate that what is exchanged is becoming more important than who is exchanging it. As an outcome, a new paradigm, called Named-Data Networking (NDN) [3, 4], has emerged as a promising future networking architecture. NDN advocates a content name-based communication model, moving from interconnecting end-points (who) to interconnecting information (what).[1] NDN provides call-by-name semantics and on-path caching as the main mechanisms to achieve high-throughput content delivery.

1.1 Content versus Services in NDN

Besides supporting static content, recent research suggests that dynamic content (i.e., services) should be supported over NDN. Several approaches arise like Service-Centric Networking [5], SOFIA [6], NFaaS [7], NextServe [8], and Named-Function Networking [9]. However, supporting services raises new challenges especially in the scope of service management in general, and load balancing in specific for four main reasons. First, services require computations while content does not. A content object (e.g., movie, document) is published by a provider and consumers request it by name. Services are different in that the content is not readily-available but needs to be prepared upon request, hence the name *dynamic content*. Second, not only does a service request require computation time, but also different requests can drastically vary in their execution times. Take for instance the following:

- SQL queries have different execution times depending on the complexity of the query.
- A service that encrypts/decrypts a file requires varying execution times depending on the size of the file and the complexity of the encryption algorithm.
- The execution times of basic algorithms (e.g., sorting, tree balancing) as well as advanced ones (e.g., machine learning algorithms) depend heavily on the input of the algorithm. Hence any service that includes such algorithms would have a large variation in its execution time.

Third, the benefits of in-network caching is limited in the case of services. Caching is an important design aspect in NDN architectures. Intermediate nodes in the network can cache content messages and future consumers can be served from these caches. When a specific content becomes popular, caching helps the network serve consumers fast and reduce the load on the original content provider. However, this is

[1] NDN belongs to a larger family of future internet architectures called Information-Centric Networking (ICN). Other ICN architectures are not in the scope of this paper.

not the case for services. Services mostly cannot be cached and even when they can, the cache is mostly useless. For instance, take a service that converts a certain media file from one format to another. It is rarely the case that the exact same service call to convert the exact same file from one format to another, is requested by several consumers within the cache lifetime. *Services* may become popular but not exact *service requests*. This is an essential difference between *content* and *services*.

Finally, the cost of a service request is not only network bandwidth as in content requests, but also CPU time, memory usage, and storage IO. This means that the original concept of broadcast forwarding strategy in NDN is extremely expensive on the service provider side. If every Interest (i.e., request in NDN) is forwarded to all possible faces (i.e., interface in NDN routers), it will reach all service replicas, each of which will execute the service request, but only one response reaches the consumer. In the presence of services, the network should make sure that a service Interest reaches only one replica, and preferably the least busiest one to be able to serve clients as fast as possible. In other words, load balancing is absolutely needed to distribute the load among service replicas in NDN.

The aforementioned differences highlight the need for load balancing in the presence of services in NDN. When the demand on a service is high, it is a standard procedure for a provider to replicate the service. In this case, there will be several service replicas in the network that provide the same service under the same name. Load balancing becomes crucial in order to distribute service requests between the replicas and avoid situations where some replicas are overloaded while others are idle. However, despite its importance, load balancing over NDN has not been studied yet.

1.2 Literature Study

In this paper, we investigate how load balancing can be designed and implemented in NDN and NDN-like architectures by surveying and analyzing closely related domains. This is not an ordinary literature review paper because of the lack of research in the area of interest. This is why we only survey domains, the concepts of which can be used for load balancing over NDN. We identify four domains as such: traditional load balancing, congestion control, forwarding strategies, and data center management over NDN. For each of these interesting domains, we identify few main studies using searches on Google Scholar, then perform Snowballing [10] to gather further studies.

Research in these domains provides valuable insights into the challenges and opportunities of load balancing. From these insights, we identify three dimensions to the design of a load balancing solution over NDN:

1. Metrics: This dimension is concerned with the metrics that would enable the load balancer to make decisions. Round-trip time, CPU load, and job queue length are examples of these metrics.
2. Purity: This dimension is concerned with the degree of compliance of a load balancing solution with NDN principles. Pure solutions transport control mes-

sages over ordinary NDN channels while hybrid solutions can use NDN and IP networks at the same time.
3. Location: This dimension is concerned with the network component that is responsible for the load balancing functionality.

Each option in each of these dimensions presents its own difficulties, opportunities, and limitations. Also a decision in one dimension can affect the options in the other dimensions.

1.3 Communicating Vessels

Three possibilities emerge out of our investigation. The first one is to place the load balancing functionality in all routers. In this case, load balancing becomes more of a forwarding strategy. The second possibility is a facade load balancer that accepts all incoming requests and then forwards them to service replicas based on pre-acquired knowledge about their state. The third possibility is to have a controller that balances job queues in service replicas in a postmortem fashion without interfering with the forwarding of requests. Evaluating the advantages and disadvantages of these scenarios, we propose and implement the *communicating vessels*, a new approach that places the load balancing functionality in the routers and uses job queue size as a metric for making forwarding decisions in order to balance the load on service replicas. Communicating vessels is implemented in PiCN [11] which adopts the basic architecture and communication model of NDN. We experimentally evaluate our new approach and show how ComVes leads to optimal load balancing in a simple network topology, and to near-optimal load balancing in a complex network topology without having to probe the network.

1.4 Contributions of the Paper

The main contributions of this paper are:

1. A survey of domains closely related to load balancing in NDN.
2. A taxonomy of the design dimensions of a load balancing approach in NDN.
3. The design, implementation, and evaluation of a new load balancing approach called the communicating vessels.

1.5 Structure of the paper

The organization of this paper follows directly the process of building a new system; from understanding the problem and evaluating potential options, to designing and implementing a new approach. After we explain the necessary concepts in NDN and motivate our work in Sect. 2, we delve into related domains in search of potentials. We survey the main approaches in load balancing over IP networks (Sect. 3), congestion control (Sect. 4), forwarding strategies (Sect. 5), and data center management in NDN (Sect. 6). In Sect. 7, we combine the insights and ideas gathered

from these related fields, and discuss the main design principles of load balancing in NDN. In Sect. 8 we use this knowledge to make informed decisions and propose the communicating vessels load balancing approach, then evaluate it over several experiments. We conclude this paper in Sect. 9 summarizing our findings and portraying future directions in this field.

2 Technical Background

Before we dive into details, it is important to summarize how NDN works and define the terms, acronyms, and concepts that are used throughout the paper. Named Data Networking [3] is an Information-Centric Networking (ICN) architecture. NDN is like any other ICN architecture, aims to solve the limitations of the current Internet architecture such as mobility and security by centering the communication around content names not locations (i.e., IPs) [12, 13]. Each piece of content is identified by a name. Providers announce their content to content routers and consumers request content by name. The main concepts of NDN are detailed by Saxena et al. [14].

The main network elements in NDN are Content routers, which forward messages between content consumers and producers in the network. Content routers contain three types of tables:

1. Content Store Table (CS), which is a cache memory that stores the retrieved data mapped with the corresponding content name.
2. Pending Interest Table (PIT), which matches between the content name and all faces that are interested to get the desired data, then the router can remember outstanding Interest faces.
3. Forwarding Information Base Table (FIB), which is a standard routing table used for Interest forwarding based on content names rather than IP addresses.

In NDN, there are two types of packets: Interest packets which are sent by consumers to get the desired data, and Data packets which are sent by producers as replies to Interests. When a router receives an Interest, it checks whether the requested content is already in the CS (i.e., cache). If yes, the router will reply by a data packet, otherwise it will check its PIT entries whether the Interest is already there or not. When there is a matching entry for the same Interest, the router adds the face of the incoming Interest in the matching PIT entry. Otherwise, the router checks the FIB table to make sure that there is a face which can serve this Interest. If the face found in FIB, the router updates the PIT table by adding a new entry for the incoming Interest with its corresponding face. Otherwise the Interest is discarded and the consumer times out.

Once the Interest reaches a node that has the requested data, this node returns a Data packet that contains the name, the content and a signature. When a router receives a Data packet with a name that has a PIT entry, it sends through the faces and deletes the PIT entry of this Interest. Then it stores the content in the CS table for further similar Interests. Data packets follow in reverse the path taken by

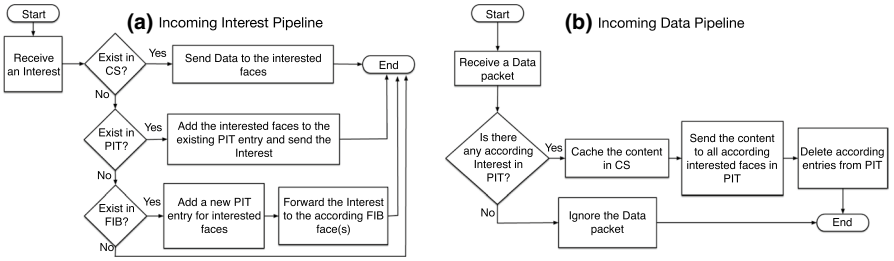

Fig. 1 NDN communication model. This figure shows the processing pipeline of incoming Interest (**a**) and Data (**b**) messages in Content routers

the Interest (breadcrumbs) to get back to the consumer. Figure 1 summarizes the procedure of processing Interest and Data packets in NDN.

3 Load Balancing in Traditional Networks

The first line of research that is highly related to load balancing in NDN is obviously load balancing in traditional IP networks. There have been numerous proposals and algorithms [15–20] but we discuss here the main foundational concepts in the field.

3.1 Round-Robin Algorithms

Round Robin (RR) is a common static load balancing algorithm that aims to limit the time for each task and fairly serve them [21]. Although the algorithm distributes the load equally between replicas, one replica can still get overloaded while the others have no load. Katevenis et al. propose weighted Round Robin algorithm (WRR) [22] to solve this issue by assigning weights to the replicas according to their capabilities. WRR requires prior knowledge about the capabilities of the replicas to set the weights and assign requests accordingly. This means that the number of assigned requests is proportional to the weights of replicas. WRR performs well only when all scheduled requests have the same size. Shreedhar et al. propose Deficit Round Robin (DRR) [23] that takes different packet sizes into consideration. However, both DRR and WRR ignore the current state of replicas. Weighted Least Connections (WLC) Round Robin is a dynamic load balancing algorithm that tackles this issue [24]. It assigns weights to replicas according to the number of current connections of each one. When the load balancer receives a new request, it forwards it to the replica with the fewest connections (weight) it has. WLC has been improved to assign weights to replicas considering replica capabilities such as processing speed and bandwidth, in addition to the number of connections that it has [25].

3.2 Load Balancing as a Scheduling Problem

Min–Min scheduling algorithm (MM) [26] estimates the minimum needed time for each job then assigns it to the node that offers a minimum of that time. Opportunistic load balancing (OLB) [27] is a static load balancing algorithm which aims to keep nodes occupied by propagating requests to the available nodes in an arbitrary order regardless of the completion time of a request at that node. OLB may cause bottlenecks as requests have to wait for a replica to be released. Load balancing Min–Min (LBMM) [28] is a dynamic load balancing algorithm which combines the Min–Min scheduling technique and the opportunistic load balancing (OLB). LBMM (dynamic) improves OLB using Min–Min algorithm, which works in a three layer architecture. The first layer is the request manager which is responsible of assigning the tasks to one of the service managers in the second layer. The service manager receives the request from the upper layer, splits the task into sub-tasks, and assigns each to a service node in the third layer based on certain metrics like CPU capacity and memory usage, which are monitored by the service manager. The service manager estimates the minimum completion time for all tasks. The task with minimum time will be chosen to be scheduled on the appropriate node according to the gathered metric information. Consequently, the completion times of all tasks that are already scheduled to be executed on this machine are updated. This procedure is repeated until all waiting tasks are scheduled. LBMM performs better when the number of small tasks is more than the number of large tasks. This approach improves the unbalanced load of Min–Min and minimizes the execution time of each node, but might lead to the starvation problem. There is another approach which uses the Max–Min algorithm [26], which is similar to Min–Min but here after computing the minimum completion time of all tasks. The priority is given to the large tasks. The main problem here is that small tasks may wait a long time to get served.

3.3 Heuristic Load Balancing

Ant Colony Optimization (ACO) [29] is a common heuristic that has been used in many load balancing algorithms [30–33]. Nishant et al. [33] propose an algorithm to balance the nodes in a cloud using ACO. At the beginning, the algorithm chooses a head node which is responsible of continuously originating ants. The head node should be the node that has the highest number of neighbors. Ants traverse the network and check the nodes' loads. The state about node resources utilization is stored in a pheromone table which is updated by ants. Once a request is issued, the ants start moving towards the destination through the nodes and recording the information about them in the pheromone table stored in each node. Ants keep going forward to check the next node load as long as the nodes they pass are underloaded. Whenever the ants find an overloaded node, they will go backward to get back to the previous node to check if it is still underloaded. If this is the case, the load will be transferred to the underloaded node. At each

movement, the pheromone table is updated to keep the network information up-to-date. The ants disappear (commit suicide) once the destination is reached.

Honeybee Behavior Load Balancing (HBB-LB) [34] is a dynamic load balancing strategy which is inspired from foraging behavior of honeybee. In HBB-LB, nodes in the cloud infrastructure are grouped into three categories: overloaded, underloaded and balanced group. Tasks are categorized as high-, medium- and low-priority. The main idea of this approach is to migrate tasks from overloaded nodes to underloaded nodes taking into consideration the priorities of tasks in the waiting queues of these replicas. These migrated tasks from overloaded nodes act as honey bees, and the underloaded nodes where tasks are moved to are considered as the destination of honey bees. Each time a task is moved to an underloaded node, it updates the priority of tasks and the node load, and this will update the number of nodes in different categories. In other words, when migrating a high priority task to another node, this node should have a minimum number of high priority tasks in its queue to guarantee that it will be served earlier. The workload of each node can be calculated based on the number of tasks at time t on service queue of node(i) divided by the service rate of node(i) at time t. The load balancing is successful when all nodes become in balanced category. The main advantage of this proposal is that it maximizes the throughput and minimizes the waiting time of tasks. However, if the number of high-priority tasks is larger than the number of low-priority ones, the latest may wait for a long time in the queue to be served. HBB-LB works only with non-preemptive tasks and considers only the priorities as QoS parameter. HBB-LB has been improved later to mitigate the original shortcomings [35, 36].

3.4 Main Takes from Traditional Load Balancing

From the work on load balancing in IP networks, there is always a special component/agent/machine that acts as the *load balancer*. The load balancing algorithm itself can be either request-based or queue-based. Request-based load balancing tries to assign new requests to the best-fit replica as in round-robin algorithms and scheduling-based algorithms. Queue-based load balancing tries to balance the job queues of replicas by migrating jobs around from one queue to another as in heuristic load balancing. Both types of load balancing rely on several metrics to take decisions, such as serve capabilities, number of current connections, and replica occupancy. These insights can be adopted in NDN networks as we discuss further in Sect. 7.

4 Congestion Control as a Load Balancing Mechanism

In general, congestion control is a mechanism applied in the transport layer to detect overloaded links and adapt network flows in order to maintain network stability, throughput, and fair resource allocation for clients. This can directly affect the loads on service providers. Congestion control can be thought of as a load balancer mechanism for network links, which in turn may lead to balanced loads on servers.

Named-Data Networking has no transport layer, but the strategy layer in the NDN stack along with the application layer provide transport functionalities. However, by design, the characteristics of the communication paradigm of NDN provides basic congestion control:

1. NDN is a pull-based system. In NDN, consumers send Interests to get the required data. For each Interest, there is at most one corresponding data packet. This receiver-driven model naturally provides traffic flow control over the network.
2. NDN routers do not propagate duplicated Interests if they are already in their PIT. Routers just aggregate faces of similar Interests in one PIT entry. Whenever a router receives a reply, it starts delivering data packet in a multicast fashion. This means that routers send a copy of the data to all aggregated faces. Propagating only one of duplicated Interests in the network reduces bandwidth usage that can be highly consumed by retransmissions. At the same time, this procedure can reduce the traffic load caused by returned data packets.
3. NDN routers have caching capability. Every time a router receives a data packet, a copy is stored in its CS. Because of this feature, future Interests for the same content will be answered from intermediate caches. Getting content from disseminated caches over the network reduces link bandwidth usage, loads on content providers, and data transmission time, thus reducing the possibility of congestions.

These concepts render the network congestion problem less important for NDN architectures than for TCP/IP networks. However, there are still certain scenarios where congestions may happen, a fact acknowledged by many researchers in the literature [52–54]. The problem is that we can not adopt TCP/IP congestion control mechanisms in NDN as it has different transport layer [55]. There are many proposals for congestion control that we can get inspiration from with respect to load balancing, which is the focus of this paper. We review and analyze some of these proposals and discuss how they link to load balancing. As illustrated by Ren et al. [56], there are three main methods for designing a congestion control algorithm in NDN: consumer-based, router-based, and hybrid methods.

4.1 Consumer-Based Congestion Control

In *consumer-based congestion control*, consumers reduce their Interest rates when a congestion is detected. This is possible because of the pull-based architecture of NDN where each Interest is satisfied by at most one data packet making consumers the initiators of communication. If Interest rate is controlled, the whole network load is controlled. There are many approaches to consumer-based congestion control [37–41, 57, 58]. In these approaches, consumers need to know when there is a

Table 1 A summary of congestion control approaches in NDN

Approach	Architecture	Based on rate or window	Detection metrics	Congestion notification
ICP [37]	Consumer-based	Window-based	Expiration timer computed from the history of RTTs	Timeout-based
ECP [38]	Consumer-based	Window-based	The queue size of Interests	Explicit notification (NACK messages)
CCTCP [39]	Consumer-based	Window-based	RTO and RTT per source	Implicit notifications
RAAQM [40]	Consumer-based	Window-based	RTO timeouts per route and RTT value per path	Timeout-based
Braun et al. [41]	Consumer-based	Window-based	RTO and RTT per CS	Timeout-based
Park et al. [42]	Router-based	Rate-based	The maximum Interest rate per face	Explicit congestion notification
Carofiglio et al. [43]	Router-based	Window-based	A separate RTO per route and the maximum number of Interests the router is allowed to send	Timeout-based
Wang et al. [44]	Router-based	Window-based	The optimal rate of each router calculated based on the bandwidth value	Hop-by-hop NACK messages
IRNA [45]	Router-based	Rate-based	The max and the min capacity of the Interest queue per face	Explicit congestion notifications
HoBHIS [46]	Router-based	Rate-based	The Interest rate calculated based to the chunk queue length	Explicit Interest rate feedback
Nguyen et al. [47]	Hybrid-based	Window-based	A congestion window and RTO value per aggregated flow (face)	Timeout-based
ChopCop [48]	Hybrid-based	Window-based from the consumer side and delay-based on the routers	The outgoing data queue length	Explicit congestion
Kato et al. [49]	Hybrid-based	Window-based	The optimal rate per flow and the RTT value between neighbors	Explicit congestion notifications
MIRCC [50]	Hybrid-based	Rate-based	The local rate and rate of the flow path	NACK messages
HR-ICP [51]	Hybrid-based	Window-based (ICP [37]) on the consumer side and rate-based on the router side	Fair share per flow per face and expiration timer caculated based on the history of RTT	Timeout-based from the consumer side

congestion and how to change the Interest rate. Table 1 shows an overview of various consumer-based congestion control approaches.

Round-trip time (RTT) for an Interest is a good indicator of the load on the network.[2] When the RTT is above a certain threshold, the consumer knows that there might be a congestion in the network for that Interest and reduces the Interest rate. This threshold is called retransmission timeout (RTO).[3] However, differently from TCP/IP networks, measuring RTTs and estimating RTOs is known to be an issue in NDN due to the caching and multi-source capabilities in the network [59, 60], i.e., A Data packet can come from different Content Stores through different network paths. To overcome this challenge, some researchers propose to maintain an RTO per CS. Braun et al. [41] propose that every CS gives each data packet it serves a unique identifier. The consumer maintains all CS identifiers seen before and stores for each one information about the served content and their RTTs. Based on this strategy, a separate RTO value is calculated and maintained for each CS. Other researchers propose to maintain an RTO per path. Carofiglio et al. propose Remote Adaptive Active Queue Management (RAAQM) [40] where each router appends its identifier to each data packet that passes by. At the end, all identifiers are aggregated to generate a route label, enabling consumers to recognize routes and to capture congestion by detecting RTO timeout for each route. In Content-Centric TCP (CCTCP) [39], consumers maintain multiple RTO values and multiple congestion windows for each flow (i.e., face).[4] One corresponds to each cache. CCTCP proposes *Anticipated Interests* to predict the location of data chunks before requesting them. This means predicting the location of future demanded chunks. Anticipating Interests enables triggering the correct RTO and maintaining a separate RTO for each anticipated source. The procedure starts from the consumer. On issuing a new Interest, the consumer must list the identifiers of the data chunks which will be requested in the close future. Each router checks in their CS if any of the chunks which will be requested in the close future exists in its CS. If this is the case, the router modifies the Interest by appending chunk identifier, a timestamp T, and a unique identifier. The same process is repeated by routers on the path till finding the content of the requested chunk. The router that has the content returns data packet, adding the anticipated chunks header retrieved from the Interest packet. CCTCP mechanism is shown in Fig. 2.

As an alternative to RTT, Interest queue size can also be an indicator of the load on the network. Ren et al. [38] propose a consumer-driven flow control protocol for NDN called Explicit Control Protocol (ECP). Routers sample the Interest queue

[2] In TCP/IP networks, the time between sending a request packet and receiving its data packet is called RTT.

[3] In TCP/IP, the requester sends a request packet and receives an acknowledgment from the receiver. If the requester realizes that there is no acknowledgement for one or more sent packets, it will stop sending requests for a duration of T then try again and send one packet to the receiver. RTO is this *waiting* time T before retrying.

[4] Congestion Window (CW) is a variable used by TCP to limit the sending rate. It is the maximum number of allowed unacknowledged packets. This window enables congestion avoidance and control. In case of congestion, the sender receives a timeout or a duplicated ACK, then it decreases the congestion window size in order to decrease the sending rate. The decrement depends on specific factor chosen by the used algorithm

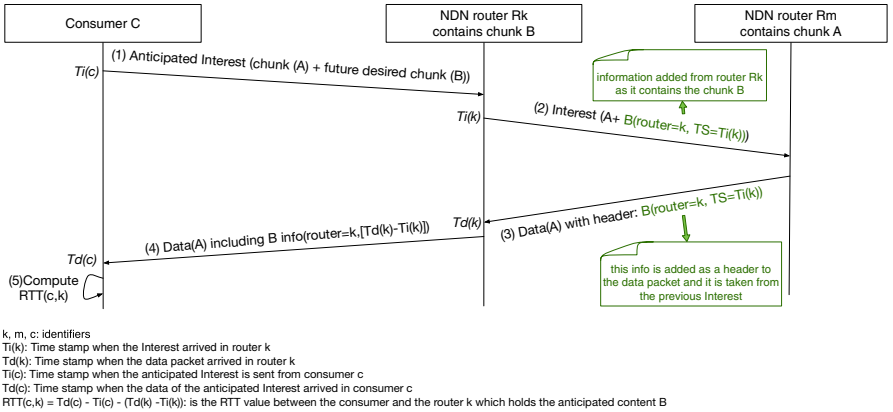

Fig. 2 This is an example of consumer-based congestion control. CCTCP [39] has the notion of anticipated Interests to predict the location of data chunks before requesting them by gathering information from the network like router identifier and RTT in each router. Then, the consumer can estimate the RTT between itself and the router which holds the anticipated data chunk

length every T time. Each sample is associated with a weight in order to compute a weighted average length of outgoing Interests, which is used to estimate the congestion status. Then, routers send a feedback to the consumer about the congestion situation in the form of Negative Acknowledgement (NACK) packets, as proposed by Yi et al. [61]. These NACK packets triggered by the consumer are: *FREE* for free links, *BUSY* for busy links, and *OVELOAD* for congested links. NACK messages are updated at each intermediate node with the heavier one compared between the original congestion level in NACK and the level detected at the node itself. At the end, the consumer gets a notification about the most congested link which is usually the bottleneck along its feedback path. Consumers adjust their sending rate of Interests according to the NACK congestion notification using Multiplicative Increase Additive Increase Multiplicative Decrease (MIAIMD) algorithm. It uses Multiplicative Increase (MI) on receiving NACK-FREE to maintain full utilization of the free bandwidth. On receiving NACK-BUSY, it uses Additive Increase (AI) in order to avoid possible congestion by smoothly increasing the congestion window size. Multiplicative Decrease algorithm (MD) is used on receiving NACK-OVERLOAD to reduce the congestion window size quickly. The main feature of ECP is that it does not relay on RTT and it provides detailed congestion status.

4.2 Router-Based Congestion Control

In *router-based congestion control*, it is the job of the routers to reduce outgoing Interest forwarding rate on different faces based on various criteria. The capability of controlling congestions by NDN router comes from the fact that routers have PIT tables containing all forwarded Interests. By monitoring the size of the PIT, routers can decide to adjust their forwarding rate and consequently control the rate of the retuned data. There are many proposals for such models [42–46, 61–63]. The main

challenge is to determine the forwarding rate according to the availability of network resources [51, 56]. Table 1 summarizes these approaches and highlights the main differences among them.

Achieving congestion control via routing strategies is the first router-driven congestion control technique. Park et al. [42] propose that the probability of choosing a forwarding face is higher for faces with shorter round-trip time RTT, where the number of Interests forwarded to a face is inverse-proportional to the RTT of receiving content through that face. Carofiglio et al. [43] propose that each router has a congestion window of Interests which defines the maximum number of outstanding Interests the router is allowed to send in order to control the Interest rate of each flow. To do so, the algorithm estimates separate RTT for every route using route-labeling [40]. RTT is monitored separately over each path from the consumer to the content producer, but there is only one aggregated congestion window that is shared by all paths and adjusted by the AIMD algorithm.[5] The algorithm monitors for each available prefix the number of PIT entries. A forwarding probability of a face is determined by a weight that is a moving average over the reciprocal count of the PIT entries. These weights are inverse-proportional to the number of pending Interest packets. The algorithm distributes incoming Interests on faces by using a weighted round robin logic. The weights are computed for all interfaces over a single prefix and then normalized across interfaces.

The second router-driven congestion control technique is based on adjusting Interest rate based on face capacity. Wang et al. [44] consider the bandwidth consumption of the Interests and corresponding data packets to limit the Interest rate in advance to maximize the link throughput in both directions. The optimal Interest shaping rate is computed based on the condition that the sum of Interest packet rate and Data packet rate should be less than the bandwidth. Otherwise, routers adjust the Interest forwarding rate based on the difference between the expected optimal rate and the actual incoming Interest rate. The limitation of this proposal comes from the assumption that link capacity is known which is unrealistic specially in wireless networks. Ahlgren adds a dynamic link capacity estimator to Wang's proposal in order to obtain high throughput and low latency [63]. Mejri et al. propose the Interest Rate Notification and Adjustment scheme (IRNA) [45], where routers monitor the number of queued Interests. For each outgoing link there are two values which determine the maximum and the minimum queue capacity. If the number of Interests is bigger than the maximum capacity, the link is congested and the router has the responsibility to determine the suitable decrease rate and which routers should decrease their sending rates. Routers must maintain multiple counters for each face. Each counter indicates the total incoming Interests via a specific face, so they can decide which input face and which prefix consumes more than its fair share. The router signals

[5] Additive-Increase/Multiplicative-Decrease algorithm (AIMD) is an algorithm used in TCP congestion control to adapt the congestion window size. In a normal situation (no congestion), AIMD increases the window size linearly by fixed values every round trip time. When congestion is captured, AIMD receives a congestion signal and multiplicatively decreases the congestion window size. AIMD aims to achieve linear growth of window size, but in case of congestion, AIMD aims to achieve exponential reduction.

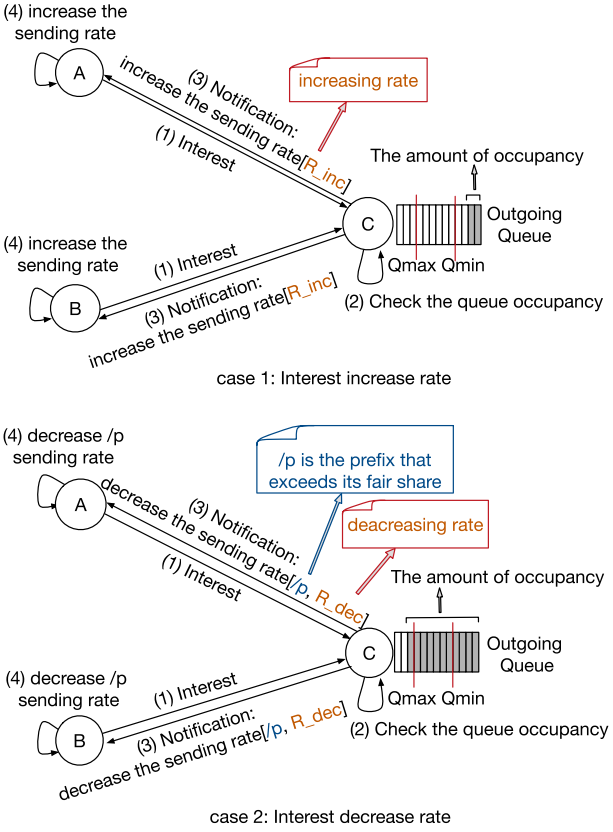

Fig. 3 This is an example of router-based congestion control. In IRNA [45], each router has a defined capacity of outgoing queue and it notifies the neighboring routers to decrease or increase their Interest rates

the decrease rate with the prefix name via its input face to all neighbors. When the number of incoming Interests is less than the minimum capacity, then the link is under-used. In this case, the router determines the suitable Interest increase rate and sends it in an explicit notification via all faces to all neighbors. When the number is between both values, the link is well used and under control, so no action is taken. Figure 3 shows the adjusting Interest rate in IRNA. The main advantage of this approach is that each router that receives a decrease notification, can first adjusts its sending rate and depending on its queue occupancy, it can decide to propagate the decrease signal or not. Consequently, in most cases the signal does not reach the consumer. Another advantage is that notifications carry out the prefix and the rate decrease in order to only control the prefixes that exceed their fair share. On the other hand, the increase rate notifications are propagated through all available faces. However, notifications are sent explicitly independently from Data packets, and this helps reacting faster in case of congestion. Rozhnova et al. propose a Hop-by-Hop Interest Shaping mechanism (HoBHIS) [46, 62] where routers compute the Interest

rate every time they receive a data chunk. This computation is calculated based on two basic parameters: the queue occupancy and the available resources (e.g., available bandwidth and the RTT value for each chunk). If the queue occupancy is smaller than a predetermined threshold, the router increases the shaping rate. Otherwise, it decreases it.

4.3 Hybrid Methods for Congestion Control

Due to the limitations of consumer-based and router-based congestion control, mixing the two methods can join their advantages and mitigate their shortcomings [47–51, 53, 54, 64–66]. Table 1 summarizes and compares these hybrid approaches.

Nguyen et al. [47] tackle the congestion control problem in NDN networks by introducing a multipath forwarding strategy to fully utilize the available bandwidth by regulating sending rate without using route-labeling. Congestion states are aggregated at the consumer nodes. This means that there will be only one aggregate flow which is accumulated by all sub-flows from consumer to sources over different routes and through the same face. In other words, there is only one congestion window for the whole aggregated flow (per face). This window is increased on receiving data packets and decreased on receiving timeouts. Congestion is detected on one of the sub-flows by tuning an adapted retransmission timeout (RTO) [67]. A data packet that is not received within a timeout is considered lost, triggering a congestion state. The congestion window is then decreased.

Chunk-switched Hop Pull Control Protocol (CHoPCoP) [48] is another congestion control mechanism which sets the size of the queue length in each intermediate node. Routers keep monitoring the size of the outgoing data queue to detect congestion. Routers use the Random Early Marking (REM) mechanism which relies on explicit congestion signaling instead of predicted RTT values. Thus, routers explicitly mark data packets and try to adjust their Interest window based on AIMD Receiver Interest Control (RIC) mechanism which consists of two phases: the slow start phase and the congestion avoidance phase which starts when the window size reaches a threshold or when the network is congested. In CHoPCoP, consumers may not cooperate. This means that REM might not effectively detect congestion and needs a hop-by-hop congestion control mechanism to assist the consumer-driven congestion control and to realize the whole congestion control. Thus, it uses a per-hop fair share Interest shaping algorithm called FISP to delay Interests at the outgoing Interest queue. Kato et al. [49] propose a different hop-by-hop congestion mechanism based on notification messages sent by routers to consumers about the optimum Interest rate that is calculated in each hop based on the link bandwidth. Consumers and routers are responsible for computing the per-hop window size to determine the optimum rate of the flow and the RTT between each two neighbors. At the end, consumers and routers determine the per-hop window size by a product of the minimum rate among the reported rates and the RTT.

Multipath-aware ICN Rate-based Congestion Control (MIRCC) [50] is another hybrid mechanism, where consumers start sending Interests at a determined initial rate. The receiver sends a data packet back carrying the maximum

rate value which is predetermined. Each forwarder computes the local rate for each link (from which it received the data packet) and compares it with the rate value carried by the data packet itself. If the local rate is smaller, the router updates the outgoing data packets with the local rate. At the end, this value will inform the consumer of the rate for the flow path in order to update its sending rate.

4.4 Congestion Control and Load Balancing in NDN

In NDN networks, estimating RTOs is not easy because NDN has a multi-source, multipath environment, which causes large variations in RTT measurements. Consumer-driven congestion control proposals in NDN consider multipath and multi-source issues by maintaining separate RTOs per flow [40], per source [39], or even per CS [41]. These solutions come with the cost of putting some overhead on consumers as they have to maintain multiple RTO values. Also, as previously shown [60, 68], RTO timeout does not give accurate information about network congestion levels. To cope with the limitations of consumer-based congestion control, researchers propose that intermediate routers should be responsible for congestion control and not consumers. However, router-based congestion control methods lead to a considerable performance overhead as all intermediate routers have extra tasks such as calculating and storing several RTOs and RTTs, and marking packets for route labeling.

In its essence, congestion control aims at controlling the load on network links to prevent packet drops and allocate network bandwidth fairly. As a side effect, we may get load balancing on the service provider side, especially with methods that prevent and control congestions by balancing the loads on the links [47, 53]. However, all the aforementioned congestion control proposals consider content producers, not service providers. Services are rarely cached and have costs other than network resources such as CPU time, storage IO, and memory usage. All these costs are not taken into consideration in congestion control solutions. Nevertheless, we can still learn from past congestion control proposals when we design a load balancing strategy:

1. It is an important decision, as in congestion control, where to implement the load balancing functionality. Putting the load balancer in the router or in the consumer would have similar advantages/disadvantages of putting congestion control solution in the router and consumer.
2. The metrics to detect network congestion can also be used to estimate the loads on service replicas. In particular, RTT and the size of the Interest queue can serve as indicators of service replica status.
3. Route-labeling and source marking are also important methods that can help network components identify the number and locations of service replicas, and aggregate load detection metrics (i.e., RTT and Interest queue size) in different ways.

Table 2 Comparison of the main proposed forwarding strategies in NDN

Approach	Metrics	Changes to NDN	Forwarding decision
MABS [87]	Retrieval time of each content per face	Pure NDN	Interests are sent to the face with the lowest retrieval time
SAF [88]	Delay, hop-count, and transmission cost	New table (FWT) to store face probabilities	Interests are sent to the best face according to their probabilities
INFORM [70]	RTT (delay of all faces)	New values (Q-values) are added to FIB.	Interests are sent to the face with the minimum delay
Stateful forwarding [61]	Smoothed RTT (SRTT)	Stores more information about face status	Interests are sent to the most highly available face with the best possible classification
EPF [73]	Face availability and RTT	Pure NDN	Interests are sent based on the weights and probabilities of faces
MDPF [72]	Interface status, pending Interest numbers	Pure NDN	Interests are sent to the face chosen based on MD method [89]
SoCCeR [69]	Provider status (CPU, memory, service load)	Pheromone tables	Interests are sent to the face with the highest probability
PAF [71]	RTT	A complete layer is added called strategy layer with special functionalities	Interests are sent to the face with the probabilistically least delay
GACF [90]	Path overhead, minimum bandwidth, RTT, and number of path hops	It splits the network into domains each is managed by an ISP in addition to the new forwarding tables	Interests are sent to the face with the highest pheromone value
PBR [91]	Content quality measured based on node capabilities and outgoing link capacity	Complex forwarding strategy that requires control messages	Interests are sent to the node with the highest quality value
DIVER [92]	Up-to-date content availability	A new DIVER FIB for probe packets	Interests are sent through the face which has the most up-to-date content
SCAN [93]	Content availability	Hybrid routing using IP routing and content routing	Interests are sent based on IP routing during heavy traffic or content availability when scanning is possible

5 Forwarding Strategies

In NDN, a forwarding strategy is responsible for choosing the best face to forward an Interest through, based on a routing protocol or according to some measurements such as link load and availability. An effective forwarding strategy improves network performance, especially when we deal with services. Choosing the best face in the FIB for a service name can reduce response time and distribute the load over service replicas. Several forwarding strategies are proposed in NDN [69–74], but we can generally put them in two main categories: strategies that rely on local information in the router and strategies that rely on network probing. In this section we review only routing strategies with the main goal of shortest response time. Other forwarding strategies that are proposed to achieve forwarding state reduction and scalability improvement [75] or better use of caching [70, 76–79] are out of the scope of this paper.[6] In Table 2 we summarize the reviewed strategies in this paper.

5.1 Local Forwarding Strategies

This family of forwarding strategies relies on information that can be collected at the router itself without relying on any external components. Many of them are already built-in strategies described in NDN architecture [3, 81, 82]. The simplest and easiest built-in forwarding strategy in NDN is the broadcast one. As the name suggests, every Interest is forwarded to all matching faces in FIB at the same time. Only the first Data packet coming from the fastest face is sent downstream. All other Data messages are dropped. This simple strategy guarantees the fastest response to clients, but creates an unnecessary redundancy of Interest and Data messages, adds an overhead on network links, and populates content caches with random content decreasing cache hits. Furthermore, in a setup where there are several service replicas in the network, each Interest is served by *all* replicas causing an overhead on server resources (e.g., CPU and memory). In fact, in such a setup, there is no point of having multiple replicas for a service. Other more advanced forwarding strategies are integrated in NDN [82]. BestRoute strategy forwards Interests to the lowest cost face defined by Named-data Link State Routing Protocol (NSLR) protocol [83] according to metrics like RTT and hop count. In this strategy, Interests are always sent to the lowest cost face till it is deleted by the routing protocol from FIB. Client Control strategy allows consumers to control where Interests go. In Adaptive Smoothed-RTT-based Forwarding (ASF) [84], Interests are sent to the faces with the lowest Smoothed RTT (SRTT)[7] and periodically probes alternative upstreams to collect SRTT measurements for unused faces. A study about the impact of some built-in NDN forwarding strategies has been done by Kalghoum et al. [85] in terms of bandwidth use, cache hit ratio, and the stability time. More approaches that are

[6] Ioannou et al. [80] and Aloulou et al. [74] have classified, evaluated, and compared forwarding strategies in terms of throughput, cache hit ratio, and Interest overhead

[7] SRTT is the predicted future round-trip times and is calculated by averaging samples of RTT over a connection.

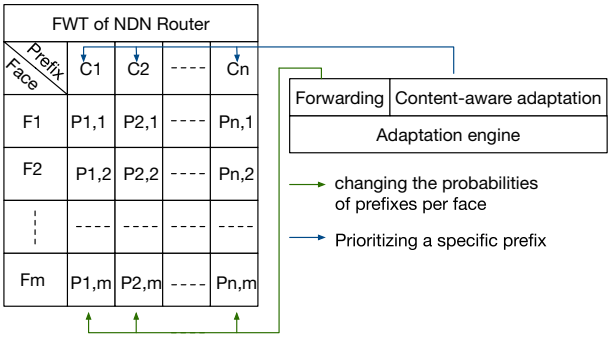

Fig. 4 An example of local forwarding strategies. In SAF [88], each router keeps an extra table of of probabilities per face per prefix. These probabilities are updated and maintained by a special adaptation engine

not integrated in NDN are still being explored and evaluated. Yi et al. [61, 86] propose an adaptive forwarding strategy that classifies faces in FIB according to their status (i.e., active if the face returns data, idle if the face is not working, or might be active or idle), and ranks them according to some metrics (e.g., SRTT). The router always chooses the highest available face in the best possible classification. NACK messages are sent back when getting no reply for Interests, which helps other nodes send Interests to other faces to explore other alternative paths.

Lei et al. propose entropy-based probabilistic forwarding (EPF) [73] to be able to select the best and most fit face for an incoming Interest. EPF considers the forwarding strategy in NDN as a *multiple attribute decision making* (*MADM*) problem [94], where the possible faces are the alternatives and different face metrics (e.g., availability and RTT) are the attributes. EPF applies the entropy theory to give weights to the face metrics, and probabilities to the faces, which are used then to forward an incoming Interest. EPF can be extended to account for any network metric. Lei et al. also propose the maximizing deviation based probabilistic forwarding (MDPF) [72], which also considers the challenge of electing the forwarding face problem as a MADM problem. The difference from EPF is that MDPF uses a Maximizing Deviation Method (MD) [89] to assign weights to attributes to dynamically react to changes in the network conditions even if they are unpredictable. Numerical experiments [72, 73] prove that both strategies EPF and MDPF have better performance than the BestRoute scheme in terms of load balancing and throughput, but both demand a considerable amount of resources on routers because forwarding an interest is not a simple table lookup anymore but rather a complicated chain of computations. Multi-armed bandits strategy (MABS) [87] aims at decreasing content retrieval time in a distributed way. A greedy algorithm is used to explore the network and build the best possible paths to forward the Interests while routers build their FIB by continuously recording the retrieval time of a specific content for each interface. On receiving a new Interest that is not cached in CS or appended to PIT, the router propagates it through all faces except the one the Interest comes through trying to classify the faces according to the retrieval time of this Interest. When a

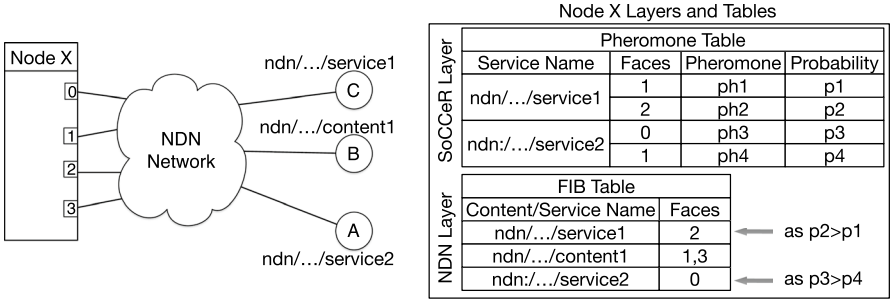

Fig. 5 An example of probing-based forwarding strategy. In SoCCeR [69], each router has an extra table called a pheromone table for services. This table contains the pheromones and the probabilities per service name per face and only the face with the highest probability goes into the FIB. Content forwarding remains untouched

router receives a similar Interest which is already initialized in its FIB, it forwards it to the face with the least mean retrieval time, and probabilistically decides whether to uniformly explore different faces, updating retrieval time for faces upon receiving Data packets.

Stochastic Adaptive Forwarding (SAF) [88] uses data packets as an input for a probability distribution which is in turn used to define the forwarding probabilities for each face per name prefix. The probability distribution is learned by maximizing a purely throughput-based measure which represents the number of satisfied Interests during a time period. These probabilities are stored in a forwarding table (FWT) which is modified by an adaption engine. Beside this, the engine gives FWT the opportunity to prioritize a specific content (prefix) as shown in Fig. 4. This priority is decided according to some statistical information.

5.2 Probing-Based Forwarding Strategies

This family of forwarding strategies requires external information from other components in the network such as other routers and service providers. This information is gathered usually via a probing mechanism.

SoCCeR [69] is a clear example of such strategies. SoCCeR acknowledges the expensive cost of redundant Interest and Data packets in NDN (in terms of network traffic, memory consumption, CPU time, and energy) when using the broadcast forwarding strategy in the presence of services. SoCCeR introduces a load balancing strategy between service replicas by incorporating Ant Colony Optimization (ACO) [29] in service routing. Routers select the best face of FIB faces according to the requested service by providing them with some parameters like available bandwidth and service load traffic. In SoCCeR, each node in the network contains a SoCCeR control layer on top of NDN layer. This layer has access to the announced services and each SoCCeR node has a pheromone table, which consists of the service name, the associated faces, the corresponding pheromone values, and the probabilities to determine the best face according to calculated pheromone values as shown in Fig. 5. Service entries in FIB are changed by adding the highest probability values

(in the pheromone table) for each face. Every service request is sent to the face with the highest probability. SoCCeR uses new types of Interest and Data packets called *Interest* and *Data Ants* to enable the functionalities of SoCCeR control layer to work. Periodically, each SoCCeR node sends an Interest Ant containing the current timestamp over all faces until reaching the target service. When the service node receives an Interest Ant, it sends back a Data Ant with a copy of the current timestamp and some information about CPU, memory and service load. Each node calculates the round trip time between itself and the service node using the status information and the time in the Data Ant. Then the pheromone value for the face on which the data arrived is updated. The face probability is calculated from the pheromones that serve as parameters for the probability formula. Currently, only two pheromones are considered in calculating the face probabilities, which are the network delay and the service replica's workload. Nevertheless, the approach allows considering additional factors easily. SoCCeR inherently implements load balancing, and routers are very responsive to service failures due to the fact that Ant Interests and Data always update pheromone tables. Li et al. [90] also uses ACO in their domain-based greedy ant colony forwarding mechanism (GACF).

Qian et al. [71] propose a probabilistic adaptive forwarding strategy (PAF), an approach similar to SoCCeR. This strategy is implemented in a new layer called the strategy layer. This layer has two main entities: the probe Engine (PrE) and the Selection Probability Processor (SPP). PrE initiates probe Interests and receives probe data in order to find the best path to the content. It also measures the RTT at each node in the path from the sender to data source and sends the latest delay to SPP. SPP computes and updates the selection probability based on statistics of the delay and the latest probed delay. PrE can use two modes to initiate probe Interests: The independent mode (IM) and the dependent mode (DM). In IM, probe and regular Interests are different and probe Interests have *Probe Identifiers*. In DM, the probe probability is not used, but regular interests are sampled and used to probe paths. The network deals with sampled interest/data similar to the regular interest/data except the sending and receiving time are recorded by PrE. However, the sampled interest/data are identified by a Probe Identifier bit as well. In this strategy, the only metric which is used to compute the selection probability is the probe delay. But there is a possibility to relay on various multiple metrics such as service load and link utilization by using a composite function of weighted probed metrics.

Eum et al. [91] introduce Potential Based Routing (PBR) as a secondary best effort routing algorithm to boost the availability of content replicas. Depending on its capability, each router decides which content needs to be defined as potential source, i.e., a node with high capacity creates a large number of potential sources and consequently serves more requests. To maintain a potential source, each node gives a quality value to each content depending on its processing CPU or the total outgoing link capacity. When receiving a request, the node compares all the quality values provided by cache neighbors and the one with the larger capacity attracts the request. PBR uses control messages to advertise content sources and exchange information them.

SCAN [93] is a scalable and on-demand routing scheme where routers are called C-routers as they perform both content routing and IP routing. C-Routers have IP

routing tables, content-based routing tables (CRT), and local content table (LCT). LCT contains all cached content at the local C-router. CRT contains all cached content at the neighbors of C-Router. Upon receiving an Interest, the router decides whether it is possible to scan neighboring routers or not according to traffic load. If it is not possible, the Interest is forwarded based on IP tables. Otherwise, the C-router updates and uses the LCT and CRT tables to find information about the requested content and sends the Interest to the corresponding face. Routers update their CRTs by exchanging them with neighbors.

Bastos et al. DIVER [92, 95] introduce Diversity-based Search and Routing mechanism (DIVER) which focuses on guaranteeing content reachability from volatile copies in the network. With a given probability, DIVER routers send out probe packets to collect information about the availability of content chunks in neighboring routers (based on a round robin mechanism), then evaluate whether the availability is up to date. If they are not up to date, routers evaluate the diversity of chunks in their DIVER FIB. DIVER only advertises cached contents on-demand to balance control traffic between popular and unpopular contents.

5.3 Tradeoffs in Forwarding Strategies

Forwarding strategies play a key role in terms of load balancing and they come with different flavors. However, each approach has its strengths and weaknesses.

Local forwarding strategies rely mainly on RTT. In case of content providers, these strategies perform reasonably well. However, when services are present, local strategies that depend on RTT face several difficulties:

1. RTT can be used when computation times on the server side are uniformly distributed, i.e., service requests trigger computations of relatively similar execution times. This is not the case for dynamic services that have varying execution times
2. RTT does not capture the state of the service provider. A large RTT indicates multiple situations: The triggered computation is costly, the server is busy with other computations, the network link is slow, or any combination of the aforementioned
3. Mixing the RTTs of service requests and data requests would induce a large variance on the statistic
4. Caching as a confounding effect on RTTs

On the positive side, local forwarding strategies do not have control messages and consequently, do not put an overhead on the network.

Probing-based strategies, on the other hand, have the potential to capture any metric from the network, from link latency to CPU and Memory usage on the server side. This makes such strategies more reliable in capturing the state of the network and making correct forwarding decisions. However, this comes at the cost of control messages like *ants* in SoCCeR [69] and *probes* in PAF [71]. These control messages traverse the network constantly, increasing traffic overhead and limiting scalability.

An important observation here is that all aforementioned strategies (except broadcasting) are computationally demanding. For each incoming Interest or Content,

many tables need to be changed to update probabilities and rankings of faces. Routing decisions are not simple anymore and require a considerable amount of resources. This puts the scalability of routers themselves to question.

Forwarding strategies can be directly used to achieve service load balancing, but need to be carefully analyzed and adapted to overcome the discussed limitations. We discuss this issue further in Sect. 7.

6 Data Center Management over NDN

There is a growing interest in the applicability of NDN architectures in data centers. Researchers acknowledge resource monitoring and load balancing as main challenges in this setup [96–101]. Although research in this area is at its preliminary state, we can still distinguish two main trends. The first trend combines IP-based and NDN-based networks in a hybrid network model. The second trend uses only NDN for data center management.

6.1 Data Centers Over Hybrid Networks

Ko et al. propose Information-Centric Data Center Network (IC_DCN) [96] to address the challenges in current data centers. In IC_DCN, the authors distinguish between control signals and data signal by placing their functionalities in two network spaces: Control plane and data plane connected over an IP-based network. In the control plane there is a logical entity called the *controller*, which regularly collects information about the topology and assign globally unique labels to each path in the topology. For multiple paths, the controller computes the shortest path trees (SPTs) rooted at each producer with all consumers at the leaves. Upon receiving a routing request, the controller applies hash-based assignment to map the request to a particular SPT rooted at a particular producer of the desired data. The main feature of IC_DCN is that it is adaptable to dynamic topologies as the controller checks the paths periodically.

In similar fashion, Chen et al. [97] propose an NDN architecture called *odICN* based on the OpenFlow SDN protocol and Datacenter technologies.[8] In odICN, the network is split into data plane and control plane as shown in Fig. 6. The data plane contains the users, data, and OpenFlow switches. The control plane contains the controller machine and three controlling algorithms: Content Locating (CL), Content Optimal Deployment (COD) and Path Optimizing (PO) algorithms. The controller applies these algorithms and update the network components in data plane via control messages over an ordinary IP network.

[8] In Software defined networking (SDN), the architecture contains a control plane which has the controller and a data plane which has the switches. OpenFlow is one of the SDN standards that define the communication protocol between the controller and the switches. The controller instruct the switches about where to send the received packets.

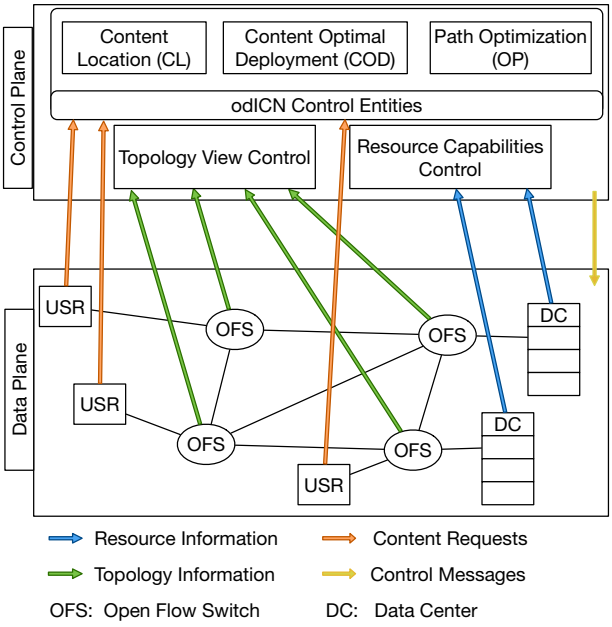

Fig. 6 An example of Data center over hybrid NDN. In odICN architecture [97]. odICN splits the network into a data plane and a control plane. The connection between both planes is based on IP. The control plane collects information about the network topology and the resource capabilities of Data centers. The controller uses this information to feed some algorithms in order to control Data centers. Users send their requests directly to the control plane

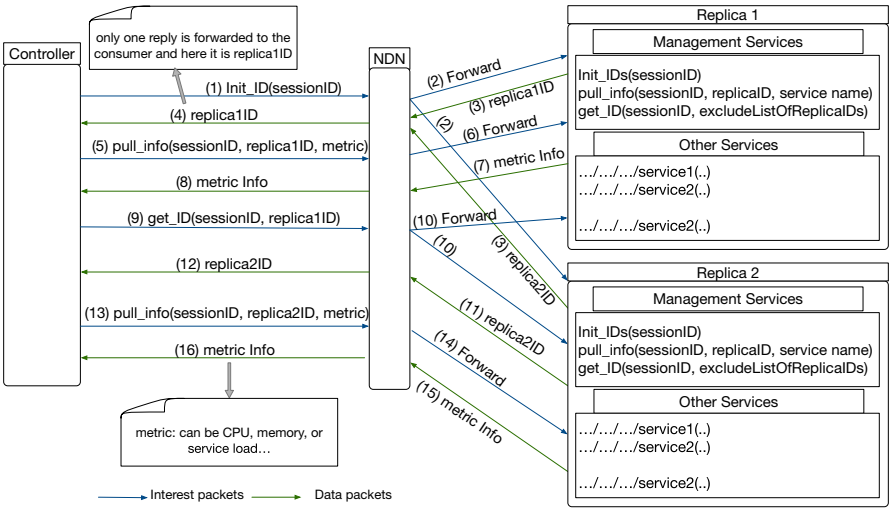

Fig. 7 An example of data center over pure NDN. In NCMP [101], the controller sends management messages to gather information about the capabilities of service replica resources. During the session, the controller gets all replica's IDs and then uses them to get information about a specific metric from a specific replica

These approaches use IP-based networks for control and NDN-based networks for data. Although it can be effective, this hybrid network model poses a high complexity on network designers and operators as they need to maintain two network stacks.

6.2 Data Centers over Pure NDN

In an earlier work, we propose a Name-Centric Management Protocol (NCMP) [101] as a monitoring protocol that works in a pure NDN. NCMP aims to scan different replicas serving the same named content or service to gather information about the load and liveliness of these replicas. NCMP can be used to monitor the performance of a data center, discover failures, and dynamically stop old replicas or initiate new replicas as needed. The idea of NCMP is to generate an ID for each replica to be able to contact a specific one by its ID. NCMP has a special entity called controller, which is responsible of sending *management* Interests to collect information about service replicas like CPU usage, memory usage, and the load on the machine. Figure 7 shows how NCMP works.

It is important to note that the management and control communication between the controller and the replicas follow the normal Interest/Data messaging paradigm. There is no control plane that operates over another network stack as in odICN [97] and IC_DCN [96]. This makes NCMP a pure NDN solution.

6.3 Insights From Data Center Management

Surveying data center management over NDN reveals two interesting aspects when it comes to load balancing. The first aspect is NDN purity. It indicates to which extent an approach is compliant with the basic NDN architecture. The second aspect is the possibility to factor out the load balancing task to an external entity like the controller in IC_DCN [96] or the monitor in NCMP [101]. We elaborate further on these aspects in Sect. 7.

7 A Taxonomy of Design Choices for Load Balancing over NDN

We identify conventional load balancing, congestion control, forwarding strategies, and data center management as the domains that are highly related to load balancing in NDN. From our review of these domains, we identify the main challenges for designing a load balancing solution for NDN and NDN-like architectures.

7.1 Metrics

This dimension is concerned with the following question: *Which metrics can be used to make load balancing decisions?*

There are several possible metrics that can be used for load balancing. However, each of which comes at a certain cost.

7.1.1 Round-Trip Time

Round-trip time is the time between sending an Interest and receiving a content. It is the sum of transmission time and execution time. As we have shown, RTT can be calculated per face, per Content Store, per path, or per Interest name. RTT per Face and per Interest name can be easily calculated locally without any modifications on the Interest or Data messages. However these RTTs are aggregations and have a large variance due to caching and multi-sourcing capabilities in NDN. On the other hand, RTT per CS and per path require route-labeling and source marking, but have finer granularity and can be more reliable. In the presence of services in the network, we need to measure the RTT per service replica to be able to estimate their current states. Thus, RTT per CS is the most appropriate if we consider that every service replica has its own CS. We can further optimize this process by applying source marking only at service replicas to limit the number of possible sources. In any case, one should keep in mind the challenges of measuring any type of RTT as discussed in Sect. 5.3.

7.1.2 Server Metrics

Collecting server metrics is helpful at estimating the states of the service replicas, and consequently identifying overloaded and underloaded ones. For instance, CPU and memory can be used to estimate the load on a certain service replica. However, collecting these metrics requires network probing. Solutions such as SoCCeR [69] and PAF [71] constantly apply network probing in all routers, introducing a considerable traffic overhead and limiting network capacity. In terms of load balancing, this probing should be done only in the network components responsible for load balancing, in a similar fashion to NCMP [101]. This way, the network is not flooded with control messages and metrics are collected only where they are consumed.

Named-Data Networking caching capabilities introduce many challenges when collecting server metrics. When the solution is pure NDN (as discussed later in Sect. 7.2), metric collection is carried out via ordinary Interest/Data communication, which is subject to caching. The network should provide a mechanism for getting the data directly from the source and not from intermediate caches. This can be done either by a special flag in the Interest message itself that tells routers to skip the CS, or by a flag in the Data message that directs routers not to cache this content.

7.1.3 Job Size

Estimating the execution time or the required resources for a service request helps the load balancer assign Interests to the right replicas. Job size metrics enable the implementation load balancing as a scheduling problem as discussed in Sect. 3.2. In some scenarios, this estimation is relatively easy, e.g., the execution time of an encryption service can be estimated based on the file size and complexity of the encryption

algorithm. In other cases, estimating the job size can be difficult, e.g., the execution time of an SQL query.

7.1.4 Queue Size

Round-trip time and server metrics enables the estimation of the *current* state of a service replica. The size of the job queue, on the other hand, enables the estimation of the load on the replica now and in the foreseen future. The queue size can be either collected from the replicas themselves via network probing or estimated locally in routers from the PIT. The first method gives an exact measure of the queue size but introduces overhead on the network. The second method can be applied locally in the routers but queue sizes are aggregated per face. Depending on where load balancing is implemented, one method or the other can be used.

7.2 NDN-Purity

This dimension is concerned with question: *To which extent is the load balancing solution compliant with NDN?* As discussed in section , this is mainly about how control messages are passed around. Control messages are network probes, server metrics, and commands to routers and service replicas. There are three possibilities: Hybrid IP/NDN network, extended NDN architecture, and pure NDN.

7.2.1 Hybrid Networks

A hybrid network solution (e.g., IC_DCN [96] and odICN [97]) splits the network into two planes: control plane and data plane. Control messages are in the control plane and are transmitted over a conventional IP network. Interests and Data messages are in the data plane and are transmitted over an NDN architecture. The main argument for this type of solutions is that it is more efficient, although this claim is a subject for further research. However, what we do know is that this hybrid network is considerably complicated because of the many network stacks and types of messages to keep track of.

7.2.2 Extended NDN Networks

An extended NDN architecture is still NDN-compliant but requires extra constructs to be able to deliver the promised functionality. SoCCeR [69] extends NDN with pheromone tables and PAF [71] introduces the strategy layer on top of NDN. However, a comprehensive evaluation of such extensions is still missing in the literature. Usually only the functions of such extensions are evaluated but we cannot foresee their impact on the main network functionality.

7.2.3 Pure NDN

In this scenario, all messages, control and data, are treated similarly and transmitted via an NDN architecture following the simple Interest/Content communication

model. This type of solutions fully adhere to the NDN architecture and relies on naming conventions rather than special constructs for communicating control messages (e.g., NCMP [101]). Such solutions are easier to implement, maintain, and interpret than the extended-NDN and Hybrid solutions.

7.3 Location

This design decision is concerned with the question: *Which network component is responsible for the load balancing functionality?*

We identify three possible locations for the load balancing functionality: The routers, a facade balancer, and a controller.

7.3.1 Load Balancing in the Routers

All routers are responsible for performing load balancing. Simply put, this is the scenario where load balancing is achieved by a forwarding strategy. The main idea is that the routers should pick the most-fit forwarding face for every Interest. The advantage of this solution is that all service replicas are available and there is no single point of failure. However, as we discussed in Sect. 5.3, this method can be demanding in terms of routers', network's, and servers' resources.

7.3.2 Load Balancing in a Facade Component

In this scenario, service replicas are shielded behind a load balancer. This load balancer accepts all incoming Interests, forwards them to replicas based on a load balancing algorithm (e.g., round robin), and passes Content replies back to consumers. In this case, the load balancer collects the metrics from the replicas and performs its own forwarding strategy to keep the load balanced on the replicas. This means that the load balancer is simply a router with a metric collection method and a special forwarding strategy. This scenario is simple, introduces minimal overhead on the network, and provides optimal load balancing. However, this solution has two main limitations. The first one is that when the load balancer crashes, the services remain offline until the load balancer recovers. The second limitation is that service replicas should be placed in special way in the network and cannot be arbitrarily distributed.

7.3.3 Load Balancing in a Controller

In this scenario, there is a network component called the controller that periodically collects metrics from service replicas and apply queue-based load balancing as discussed in Sect. 3.4. Service replicas receive Interests normally via the network forwarding strategy. Then the controller, with the necessary metrics collected similarly to NCMP [101], commands the overloaded replicas to forward interests in their queue to underloaded replicas. This method requires that the controller should have a method for identifying service replicas and collecting metrics from them. Also this method requires overloaded replicas to be able to redirect interests to other specific

replicas that are underloaded. This approach is more complicated than the previous two, but allows for service replicas to be placed anywhere in the network and has no single point of failure. If the controller is down, service replicas remain active and serve consumers normally. When the controller is up again, it can perform post-mortem load balancing.

8 Communicating Vessels: A Load Balancing Approach

After analyzing the possible options from the three aforementioned dimensions, we propose a load balancing approach where the metric is *queue size*, the location is *routers*, and the approach is an *extended NDN* with an extra construct. The main idea is that if routers balance the loads on their faces, the loads on service replicas are also balanced. We call our approach the *Communicating Vessels* (ComVes). The analogy would be that service replicas are the vessels, routers are the connecting base, and load is the water.

8.1 System Design

8.1.1 Extending the PIT

The only change that is required to NDN is that the *Pending Interest Table* (*PIT*) should contain the outgoing face for each of its entries. In an ordinary PIT, each entry simply contains the interested face and the Interest. We add an extra column, called outgoing face, containing the face that the Interest was forwarded to.

8.1.2 Queue Size

After extending the PIT with the outgoing face information, we can compute the number of pending requests per face, per service name. For instance, take the following scenario:

- The name *ch/unibas/myService* can be served on the faces 1, 3, and 5.
- The router received 10 Interests for this service; *ch/unibas/myService/{x1 ... x10}*.
- Three of these Interests are forwarded to face 1, six Interests to face 3, and one Interest is forwarded to face 5.

We can then compute the *Busyness* of faces with respect of the service *ch/unibas/myService* as: face 1 has three pending Interests, face 3 has six pending Interests, and face 5 has one pending Interests. The number of pending service Interests per face acts as the queue size of the service replica(s) that hide behind that face. In this sense, there may be any number of service replicas, but as far as the router is concerned, there are three queues that need to be balanced. We call the *Busyness* of faces with respect to a certain service, *the Busyness table*, which can be

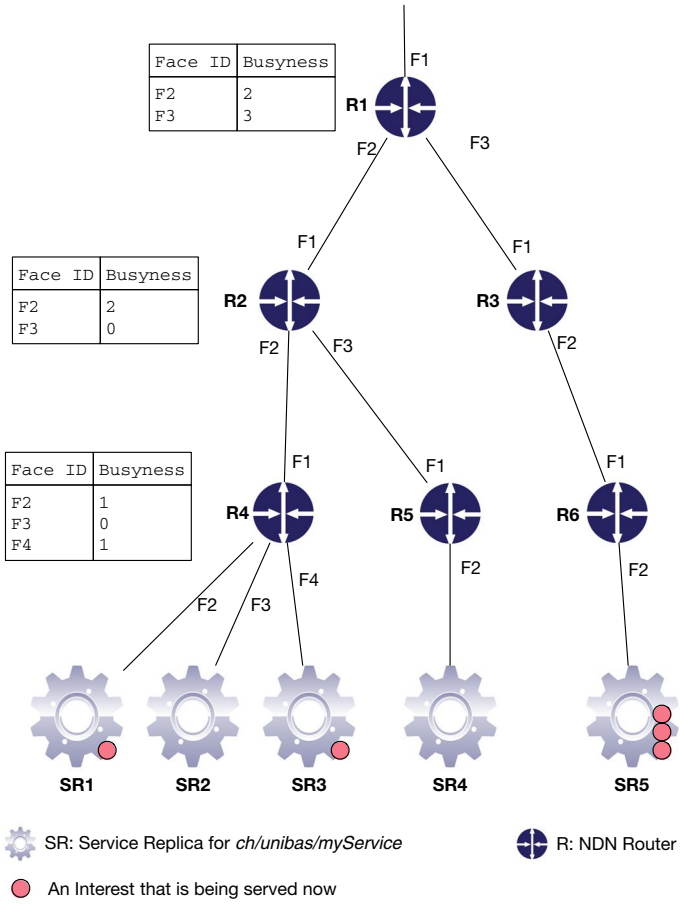

Fig. 8 An example network constellation with *Busyness* tables. Each router "sees" different number of service queues. R1 sees only two, but one of them gets served four times faster than the other because there are four service replicas behind one face and only one replica behind the other. R4 on the other hand "sees" three equally fast-served queues. According to the communicating vessels algorithm, the next Interest arriving in R1 will be forwarded to either SR2 or SR4

calculated on demand. Figure 8 shows a network constellation with the *Busyness* table on each router with respect to the service *ch/unibas/myService*. The *Busyness* table needs to be calculated on every incoming Interest with regard to the requested service name. A face with the number of 10 outgoing Interests, none of which is for the requested service, would have the *Busyness* value of 0 for that exact service. This process could incur processing overhead in routers with large PIT tables, with more than one face for the requested service, and with links to more than one service. In routers that have one face for the requested service, there is no need for the calculation at all. In routers that forward to only one service name (e.g., facade routers in front of server cluster) can maintain a simple

table with the number of pending Interests per face. Otherwise, the calculation of the *Busyness* table on each incoming Interest is mandatory.

8.1.3 The Algorithm

When an Interest arrives in a router, and exactly at the point when a certain face needs to be chosen from a list of possible faces (according to the FIB), the router calculates the *Busyness* of each of the faces with respect to the service name. The calculation of *Busyness* tables is triggered for each Interest to reflect the current state of faces. Then the router sends the Interest to the face with the least number of outgoing Interests. When every router applies this simple routing method, the load on service replicas is balanced. Figure 8 shows a network constellation and four service replicas for the service *ch/unibas/myService*. Service replica 5 (SR5) is processing three requests, while SR1 and SR3 are serving one request each. SR2 and SR4 are idle. Following the ComVes algorithm, the next Interest coming to the router R1 will be directed all the way towards SR4. Assuming that these service replicas are identical, R1 sees only two queues: Face 2 (F2) and Face 3 (F3). However, if a 100 Interests for *ch/unibas/myService* arrive to R1, it will ideally forward 80 of them towards F2 and 20 towards F3 because the queue on F2 will be served four times faster than the queue on F3. The router does not know about the replicas behind each face, but the *Busyness* table will reflect that reality and allow the router to make the best forwarding choice. In the evaluation section, we empirically show how close ComVes gets to this ideal load balancing.

ComVes combines the simplicity of placing a load balancing mechanism in the router with the accuracy of the queue size as a metric, without core changes in NDN architecture and without loading the network with probe messages. Table 3 compares the ComeVes with major surveyed approaches regarding the three design dimensions.

8.2 Empirical Evaluation

We have implemented ComVes on top of PiCN [11]. In the following section, we carry out several experiments to evaluate our approach and reveal its strengths and weaknesses.

In this section, we empirically answer these two main questions:

1. RQ1: Does ComVes reduce service response time when we add more service replicas?
2. RQ2: How does ComVes perform when router faces are unbalanced in terms of the number of replicas behind?

8.2.1 Communicating Vessels in a Simple Topology

In this evaluation, we evaluate the benefit of applying ComVes on interest response time and on server load. For this experiment, we design a simple

Table 3 Comparison of the communicating vessels with the major surveyed approaches with the respect to the three design dimensions: Metric, NDN-purity, and location

Approach	Metric			NDN Purity			Location		
	RTT	Queue size	Server capabilities	No change or minor change	Major change	Hybrid solution	Router	Consumer	Controller
SoCCeR			✓		✓		✓		
PAF	✓				✓		✓		
GACF	✓		✓		✓		✓		
SAF			✓		✓		✓		
INFOFRM	✓			✓			✓		
EPF	✓		✓	✓			✓		
IRNA		✓			✓		✓		
ICP	✓			✓				✓	
ECP		✓		✓				✓	
ChopCop	✓			✓			✓	✓	
HR-ICP	✓			✓			✓	✓	
IC-DCN			✓			✓			✓
NCMP			✓	✓					✓
PBR			✓			✓	✓		
ComVes		✓		✓			✓		

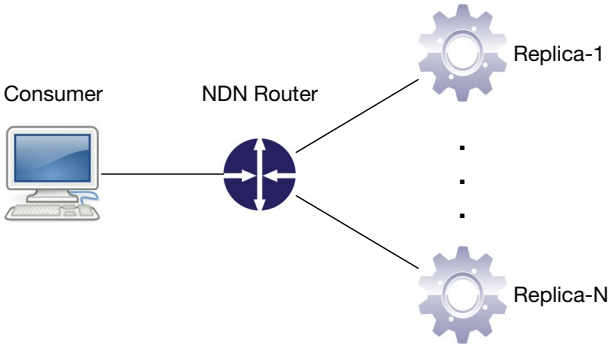

Fig. 9 This is a simple network topology where the NDN router is connected to three replicas

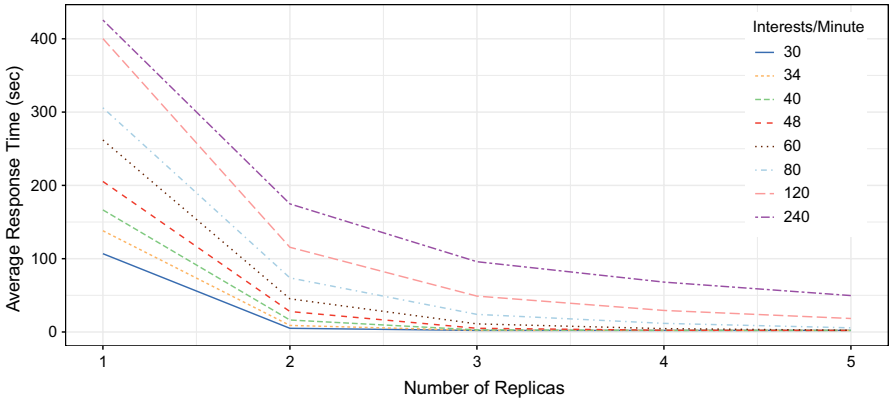

Fig. 10 This experiment is carried out on the simple topology described in Fig. 9. It shows the relationship between the average response time and the number of replicas, under several load conditions. It shows that adding more replicas reduces the response time as expected

network topology as in Fig. 9, where the consumer is connected to an NDN router which is connected to one or more service replicas. Each of the service replicas is a MacBook Pro with a processor 2.5 GHz Intel core i7 and 16 GB of memory. Each of the NDN router and the consumer runs on MacBook Pro with a processor 2.2 GHz Intel core i7 and 16 GB of memory. The service provided by these replicas is the Bernoulli function which takes an integer n and computes its Bernoulli number [102, 103]. We choose this function because it is a non-trivial calculation that takes increasingly longer to compute as the parameter goes higher.

In this experiment, the consumer sends Interests to the Bernoulli service with varying parameter values at varying intervals:

– Service parameter belongs to the range [200, 700]. This means that there are 500 Interests per experiment.

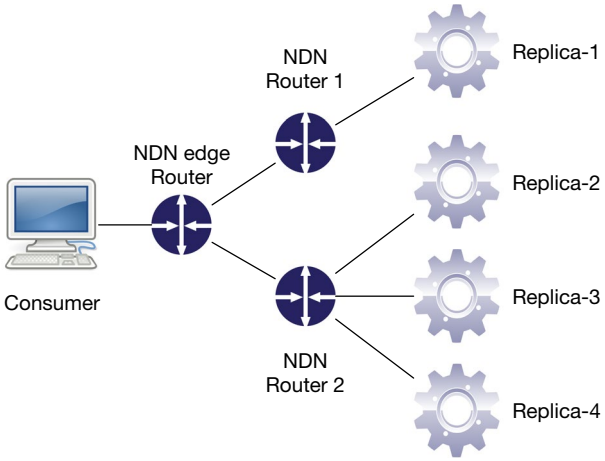

Fig. 11 This is a complex topology where the NDN edge router has two faces for the Bernoulli service. However, one face leads to one service replica, while the other leads to four replicas

– We run the experiment several times with varying intervals between Interests. The interval ranges between 0.25 and 2 s, with a step of 0.25 s. This results in different Interest rates per minute.
– We collect the statistics about the response time for each Interest and the load on each replica.

Figure 10 shows the results of the experiment. As the diagram shows, adding more replicas significantly reduces the response time as the load gets split among the available replicas depending on the *Busyness* table in the NDN router. This answers our first research question (RQ1). ComVes also makes sure that each Interest is handled by one replica only whereas in the default broadcast strategy, all available replicas would be working on the same Interests at the same time. This means that not only the response time is reduced on the consumer side, but also producer resources are better utilized and redundant work is eliminated.

8.2.2 Communicating Vessels in a Complex Topology

In the previous experiment, the topology is simple, the replicas are identical, and router faces are equivalent in terms of the number of connected replicas, i.e., one replica per face. However, in a complex network topology, there might be imbalance in the number of replicas behind each face. In this experiment, we evaluate such scenario on a complex topology as in Fig. 11. More concretely, we are interested in

1. whether the load on the two faces of the NDN edge router is equal or is shifted towards the face with more computing power behind,
2. whether the average response time is affected by this complex topology or not.

Table 4 This table shows the load (number of Interests) on the two unbalanced faces of the edge router in Fig. 11 under different conditions

Parameter range	Interest interval (s)	3-Replica face load	1-Replica face load
1 to 1000	0.25	399	601
	0.10	360	640
501 to 1500	0.25	374	626
	0.10	329	671

The results indicate that ComVes is not optimal, but is still sensitive to the computation power behind each face. Also the higher the load on the service, the better ComVes distributes the results

Table 5 A comparison between the average response times with four replicas in simple and complex topologies

Topology	Interest interval (s)	Average response time (s)
Simple	0.25	68.0
Complex	0.25	84.9
Simple	0.50	29.3
Complex	0.50	37.0

In the complex topology in Fig. 11, each replica is a MacBook Pro with a processor 2.5 GHz Intel core i7 and 16 GB of memory. Each of the NDN routers and the consumer runs on a MacBook Pro with a processor 2.2 GHz Intel core i7 and 16 GB of memory. The NDN edge router has two faces. The first face is connected to Router 1, which is connected to only one replica. We call it the weak face or the 1-replica face. The other face is connected to Router 2, which leads to three replicas. We call it the 3-replica face or the strong face. To evaluate the distribution of interests among unbalanced faces, we run four experiments that differ in the Bernoulli service parameter range and the interval between Interests. Table 4 shows the load (number of forwarded Interests) on each of the faces in the NDN edge router. We observe that the edge router does not split the Interests equally between the faces, but rather forwards more than 60% of the Interests to the strong face. This means that ComVes is sensitive to the computation power behind each face, but not optimally. In an optimal distribution 75% of the Interests would be forwarded to the strong face. However, we observe that the higher the load on a service, the closer the distribution to optimum. When the service parameter ranges between 501 and 1500, the Bernoulli service takes more time and computation power than when the parameter ranges between 1 and 1000. Also when the interval between Interests is 0.1 s, the load in the service replicas becomes higher than when the interval is 0.25 s. When the interval is 0.1 s and the service parameter belongs to [501, 1500], the load on the strong face reaches 67%, which is closer to the optimal value of 75%.

This near-optimal distribution of Interests between the weak and strong face, can affect the response time. We compare the average response time between the

complex topology in Fig. 11 and in the simple topology with four replicas as in Fig. 9 under the following setups:

- The consumer sends 500 Interests for the Bernoulli service with the parameter ranging from 201 to 700.
- High load of Interests with an interval of 0.25 and 0.5

As expected, Table 5 shows that the face imbalance leads to an increase between 24.9 and 26.5% on the average response time. Similarly to the previous observation, we also see that the higher the load on the service, the better ComVes performs.

8.2.3 Evaluation Summary

The results in Tables 4 and 5 show that a complex network topology leads to an only near-optimal load balancing with ComVes. However, routers with ComVes forwarding strategy can still sense the computation power behind each face and adapt Interest forwarding accordingly, without any network probing or control messages. As with any approach, there are always pros and cons, and network designers and engineers need to be aware of the tradeoffs at hand.

In summary, ComVes has the following properties:

- It requires minimal changes to NDN routers i.e., only one column is added to the PIT table.
- It is computationally cheap on the router side i.e., only one simple calculation before forwarding Interests is needed.
- Optimal load balancing in simple network topologies.
- It requires no network probing or control messages.
- Near-optimal load balancing in complex network topologies, but with a trend towards optimality as the demand on the service increases.
- It does not rely on RTT, and thus avoids its limitations [59, 60].

It is worth noting that the carried evaluation employs small-sized network topologies. A more realistic assessment of the proposed approach would require further experiments with appropriately-sized topologies that reflect real world networks.

9 Conclusions

Scalability is essential for modern IT infrastructure. Service providers strive to serve increasing numbers of users. Named-Data Networking (NDN) as a future Internet architecture is, by design, a scalable solution for content distribution. The more demand on a piece of content, the more responsive the network becomes due to its on-path caching capability. However, services (i.e., dynamic content) do not benefit from the strength of NDN, and thus require replication and load balancing in order to scale. Nevertheless, load balancing is an overlooked topic in the field of NDN.

In this paper, we design a load balancing solution in NDN by surveying closely related fields and uncovering what can and cannot be adopted. We identify three main design dimensions to consider before designing a load balancing functionality. The first dimension is concerned with the metrics that can be used by a load balancer in order to make decisions. Round-trip time, server metrics, job size, and queue size are the main identified metrics. The second dimension considers the compliance with NDN principles. A load balancing solution can be either pure or hybrid with respect to NDN. The third dimension is the location of the load balancing functionality, which can be the routers, a facade component in front of service replicas, or a controller that balances the load in a post-mortem fashion. We discuss the interplay among these dimensions and how a choice in one affects the choices in the others. This study can guide researchers and practitioners through the design and implementation of a load balancing solution to scale their services and make use of the potentials of NDNs. We finally implement and evaluate a new load balancing approach called communicating vessels (ComVes), which combines the pros of the studied design dimensions while avoiding their cons. We empirically show how ComVes achieves load balancing in simple and complex network topologies and manages to detect the best faces to forward incoming Interests without having to probe the network. In the future, we plan to improve our approach to tackle its imperfection in complex network topologies.

References

1. Blumenthal, Marjory S., Clark, David D.: Rethinking the design of the internet: end to end arguements vs. the brave new world. ACM Trans. Internet Technol. **1**, 70–109 (2001)
2. Papadimitriou, D., Zahariadis, T., Martinez-Julia, P., Papafili, I., Morreale, V., Torelli, F., Sales, B., Demeester, P.: Design principles for the future internet architecture. In: Federico, Á., Frances, C., Petros, D., John, D., Alex, G., Ana, G., Anastasius, G., Stamatis, K., Srdjan, K., Man-Sze, L., Volkmar, L., Henning, M., Elio, S., Anne-Marie, S., Hans, S., Burkhard, S., Georgios, T., Petra, T., Theodore, Z. (eds.) The future internet, pp. 55–67. Springer, Berlin (2012)
3. Van Jacobson, Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. In: Proceedings of the 5th international conference on emerging networking experiments and technologies, CoNEXT '09, pp. 1–12. ACM, New York, NY, USA (2009)
4. CCN Lite: Lightweight implementation of the content centric networking protocol. http://ccn-lite.net/ (2013). Accessed 23 Jan 2019
5. Braun, T., Hilt, V., Hofmann, M., Rimac, I., Steiner, M., Varvello, M.: Service-centric networking. In: 2011 IEEE international conference on communications workshops (ICC), pp. 1–6 (June 2011)
6. Wu, Q., Li, Z., Zhou, J., Jiang, H., Hu, Z., Liu, Y., Xie, G.: Sofia: toward service-oriented information centric networking. IEEE Netw. **28**(3), 12–18 (2014)

7. Król, M., Psaras, I.: Nfaas: named function as a service. In: Proceedings of the 4th ACM conference on information-centric networking, ICN '17, pp. 134–144, ACM, New York, NY, USA (2017)
8. Mansour, D., Braun, T., Anastasiades, C.: Nextserve framework: supporting services over content-centric networking. In: Wired/wireless internet communications, pp. 189–199. Springer International Publishing, Cham (2014)
9. Tschudin, C., Sifalakis, M.: Named functions and cached computations. In: 2014 IEEE 11th consumer communications and networking conference (CCNC), pp. 851–857 (Jan 2014)
10. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, EASE '14, pp. 38:1–38:10. ACM, New York, NY, USA (2014)
11. PiCN: A modular and extensible library for Content-Centric Networking (2018). Accessed 31 Mar 2019
12. Pan, J., Paul, S., Jain, R.: A survey of the research on future internet architectures. IEEE Commun. Mag. **49**(7), 26–36 (2011)
13. EC FIArch Group: Fundamental limitations of current internet and the path to future internet (2011)
14. Saxena, Divya, Raychoudhury, Vaskar, Suri, Neeraj, Becker, Christian, Cao, Jiannong: Named data networking: a survey. Comput. Sci. Rev. **19**, 15–55 (2016)
15. Nuaimi, K.A., Mohamed, N., Nuaimi, M.A., Al-Jaroodi, J.: A survey of load balancing in cloud computing: challenges and algorithms. In: 2012 second symposium on network cloud computing and applications, pp. 137–142 (Dec 2012)
16. Aslam, S., Shah, M.A.: Load balancing algorithms in cloud computing: a survey of modern techniques. In: 2015 National software engineering conference (NSEC), pp. 30–35 (Dec 2015)
17. Ivanisenko, I.N., Radivilova, T.A.: Survey of major load balancing algorithms in distributed system. In: 2015 information technologies in innovation business conference (ITIB), pp. 89–92 (Oct 2015)
18. Ghomi, Einollah Jafarnejad, Rahmani, Amir Masoud, Qader, Nooruldeen Nasih: Load-balancing algorithms in cloud computing: a survey. J. Netw. Comput. Appl. **88**, 50–71 (2017)
19. Thakur, Avnish, Goraya, Major Singh: A taxonomic survey on load balancing in cloud. J. Netw. Comput. Appl. **98**, 43–57 (2017)
20. Zhang, J., Yu, F.R., Wang, S., Huang, T., Liu, Z., Liu, Y.: Load balancing in data center networks: a survey. IEEE Commun. Surv. Tutor. **20**(3), 2324–2352 (2018). **(thirdquarter)**
21. Mohapatra, Subasish, Mohanty, Subhadarshini, Smruti Rekha, K.: Analysis of different variants in round robin algorithms for load balancing in cloud computing. Int. J. Comput. Appl. **69**(22), 17–21 (2013)
22. Katevenis, M., Sidiropoulos, S., Courcoubetis, C.: Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. IEEE J. Sel. Areas Commun. **9**(8), 1265–1279 (1991)
23. Shreedhar, M., Varghese, G.: Efficient fair queuing using deficit round-robin. IEEE/ACM Trans. Netw. **4**(3), 375–385 (1996)
24. Choi, D.J., Chung, K.S., Shon, J.: An improvement on the weighted least-connection scheduling algorithm for load balancing in web cluster systems. In: Kim, T.H., Yau, S.S., Gervasi, O., Kang, B.-H., Stoica, A., Siezak, D. (eds.) Grid and distributed computing, control and automation, pp. 127–134. Springer, Berlin (2010)
25. Ren, X., Lin, R., Zou, H.: A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. In: 2011 IEEE international conference on cloud computing and intelligence systems, pp. 220–224 (Sept 2011)
26. Ibarra, Oscar H., Kim, Chul E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors. J. ACM **24**(2), 280–289 (1977)
27. Braun, T.D., Siegel, Howard Jay, Beck, Noah, Bölöni, Ladislau L., Maheswaran, Muthucumaru, Reuther, Albert I., Robertson, James P., Theys, Mitchell D., Yao, Bin, Hensgen, Debra, Freund, Richard F.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **61**(6), 810–837 (2001)
28. Wang, S.-C., Yan, K.Q., Liao, W.-P., Wang, S.-S.: Towards a load balancing in a three-level cloud computing network. In: 2010 3rd international conference on computer science and information technology, vol. 1, pp. 108–113 (July 2010)
29. Dorigo, M., Birattari, M., Stutzle, T.: Ant Colony Optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)

30. Li, K., Xu, G., Zhao, G., Dong, Y., Wang, D.: Cloud task scheduling based on load balancing Ant Colony Optimization. In: 2011 sixth annual chinagrid conference, pp. 3–9 (Aug 2011)
31. Hsiao, Y.-T., Chuang, C.-L., Chien, C.-C.: Computer network load-balancing and routing by Ant Colony Optimization. In: Proceedings. 2004 12th IEEE international conference on networks (ICON 2004) (IEEE Cat. No.04EX955), vol. 1, pp. 313–318 (Nov 2004)
32. Ragmani, A., El Omri, A., Abghour, N., Moussaid, K., Rida, M.: A performed load balancing algorithm for public cloud computing using Ant Colony Optimization. In: 2016 2nd International conference on cloud computing technologies and applications (CloudTech), pp. 221–228 (May 2016)
33. Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K.P., Nitin, Rastogi, R.: Load balancing of nodes in cloud using Ant Colony Optimization. In: 2012 UKSim 14th international conference on computer modelling and simulation, pp. 3–8 (March 2012)
34. Dhinesh Babu, L.D., Venkata Krishna, P.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. **13**(5), 2292–2303 (2013)
35. Kaur, A., Kaur, B.: Load balancing in tasks using honey bee behavior algorithm in cloud computing. In: 2016 5th international conference on wireless networks and embedded systems (WECON), pp. 1–5 (Oct 2016)
36. Sheeja, Y.S., Jayalekshmi, S.: Cost effective load balancing based on honey bee behaviour in cloud environment. In: 2014 first international conference on computational systems and communications (ICCSC), pp. 214–219 (Dec 2014)
37. Carofiglio, G., Gallo, M., Muscariello, L.: ICP: design and evaluation of an interest control protocol for content-centric networking. In: Computer communications workshops (INFOCOM WKSHPS), 2012 IEEE conference on, IEEE, pp. 304–309 (2012)
38. Ren, Y., Li, J., Shi, S., Li, L., Wang, G.: An explicit congestion control algorithm for named data networking. In: 2016 IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 294–299 (April 2016)
39. Saino, L., Cocora, C., Pavlou, G.: CCTCP: a scalable receiver-driven congestion control protocol for content centric networking. In: 2013 IEEE international conference on communications (ICC), pp. 3775–3780 (June 2013)
40. Carofiglio, G., Gallo, M., Muscariello, L., Papali, M.: Multipath congestion control in content-centric networks. In: 2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 363–368 (April 2013)
41. Braun, S., Monti, M., Sifalakis, M., Tschudin, C.: An empirical study of receiver-based AIMD flow-control strategies for CCN. In: 2013 22nd international conference on computer communication and networks (ICCCN), pp. 1–8 (July 2013)
42. Park, H., Jang, H., Kwon, T.: Popularity-based congestion control in named data networking. In: 2014 sixth international conference on ubiquitous and future networks (ICUFN), pp. 166–171 (July 2014)
43. Carofiglio, G., Gallo, M., Muscariello, L., Papalini, M., Wang, S.: Optimal Multipath congestion control and request forwarding in information-centric networks. In: 2013 21st IEEE international conference on network protocols (ICNP), pp. 1–10 (Oct 2013)
44. Wang, Y., Rozhnova, N., Narayanan, A., Oran, D., Rhee, I.: An improved hop-by-hop interest shaper for congestion control in named data networking. In: Proceedings of the 3rd ACM SIGCOMM workshop on information-centric networking, ICN '13, pp. 55–60, ACM, New York, NY, USA (2013)
45. Mejri, S., Touati, H., Kamoun, F.: Hop-by-hop interest rate notification and adjustment in named data networks. In: 2018 IEEE wireless communications and networking conference (WCNC), pp. 1–6 (April 2018)
46. Rozhnova, N., Fdida, S.: An effective hop-by-hop interest shaping mechanism for CCN communications. In: 2012 proceedings IEEE INFOCOM workshops, pp. 322–327 (March 2012)
47. Nguyen, D., Fukushima, M., Sugiyama, K., Tagami, A.: Efficient multipath forwarding and congestion control without route-labeling in CCN. In: 2015 IEEE international conference on communication workshop (ICCW), pp. 1533–1538 (June 2015)
48. Zhang, F., Zhang, Y., Reznik, A., Liu, H., Qian, C., Xu, C.: A transport protocol for content-centric networking with explicit congestion control. In: 2014 23rd international conference on computer communication and networks (ICCCN), pp. 1–8 (Aug 2014)
49. Kato, T., Bandai, M.: A hop-by-hop window-based congestion control method for named data networking. In: 2018 15th IEEE annual consumer communications networking conference (CCNC), pp. 1–7 (Jan 2018)

50. Mahdian, M., Arianfar, S., Gibson, J., Oran, D.: MIRCC: multipath-aware ICN rate-based congestion control. In: Proceedings of the 3rd ACM conference on information-centric networking, ACM-ICN '16, pp. 1–10, ACM, New York, NY, USA (2016)
51. Carofiglio, G., Gallo, M., Muscariello, L.: Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In: Proceedings of the second edition of the ICN workshop on information-centric networking, ICN '12, pp. 37–42, ACM, New York, NY, USA (2012)
52. Tanaka, D., Kawarasaki, M.: Congestion control in named data networking. In: 2016 IEEE international symposium on local and metropolitan area networks (LANMAN), pp. 1–6 (June 2016)
53. Schneider, K., Yi, C., Zhang, B., Zhang, L.: A practical congestion control scheme for named data networking. In: Proceedings of the 3rd ACM conference on information-centric networking, ACM-ICN '16, pp. 21–30, ACM, New York, NY, USA (2016)
54. Ndikumana, A., Ullah, S., Thar, K., Tran, N.H., Ju Park, B., Hong, C.S.: Novel cooperative and fully-distributed congestion control mechanism for content centric networking. IEEE Access **5**, 27691–27706 (2017)
55. Chen, Q., Xie, R., Yu, F.R., Liu, J., Huang, T., Liu, Y.: Transport control strategies in named data networking: a survey. IEEE Commun. Surv. Tutor. **18**(3), 2052–2083 (2016). **(thirdquarter)**
56. Ren, Yongmao, Li, Jun, Shi, Shanshan, Li, Lingling, Wang, Guodong, Zhang, Beichuan: Congestion control in named data networking: a survey. Comput. Commun. **86**, 1–11 (2016)
57. Salsano, S., Detti, A., Cancellieri, M., Pomposini, M., Blefari-Melazzi, N.: Transport-layer issues in information centric networks. In: Proceedings of the second edition of the ICN workshop on information-centric networking, ICN '12, pp. 19–24, ACM, New York, NY, USA (2012)
58. Amadeo, M., Molinaro, A., Campolo, C., Sifalakis, M., Tschudin, C.: Transport layer design for named data wireless networking. In: 2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 464–469 (April 2014)
59. Xu, Y., Yao, S., Wang, C., Xu, J.: CO-RTO: achieving efficient data retransmission in VNDN by correlations implied in names. In: 2017 IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 366–371 (May 2017)
60. Psaras, Ioannis, Tsaoussidis, Vassilis: Why TCP timers (still) don't work well. Comput. Netw. **51**(8), 2033–2048 (2007)
61. Yi, Cheng, Afanasyev, Alexander, Moiseenko, Ilya, Wang, Lan, Zhang, Beichuan, Zhang, Lixia: A case for stateful forwarding plane. Comput. Commun. **36**(7), 779–791 (2013)
62. Rozhnova, N., Fdida, S.: An extended hop-by-hop interest shaping mechanism for content-centric networking. In: 2014 IEEE global communications conference, pp. 1–7 (Dec 2014)
63. Ahlgren, B., Hurtig, P., Abrahamsson, H., Grinnemo, K., Brunstrom, A.: ICN congestion control for wireless links. In: 2018 IEEE wireless communications and networking conference (WCNC), pp. 1–6 (April 2018)
64. Zhong, S., Liu, Y., Li, J., Lei, K.: A rate-based multipath-aware congestion control mechanism in named data networking. In: 2017 IEEE international symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ISPA/IUCC), pp. 174–181 (Dec 2017)
65. Zhou, J., Wu, Q., Li, Z., Kaafar, M.A., Xie, G.: A proactive transport mechanism with explicit congestion notification for NDN. In: 2015 IEEE international conference on communications (ICC), pp. 5242–5247 (June 2015)
66. Zhang, Feixiong, Zhang, Yanyong, Reznik, Alex, Liu, Hang, Qian, Chen, Chenren, Xu: Providing explicit congestion control and multi-homing support for content-centric networking transport. Comput. Commun. **69**(C), 69–78 (2015)
67. Jacobson, V.: Congestion avoidance and control. In: Symposium proceedings on communications architectures and protocols, SIGCOMM '88, pp. 314–329, ACM, New York, NY, USA (1988)
68. Zhang, L.: Why TCP timers don't work well. SIGCOMM Comput. Commun. Rev. **16**(3), 397–405 (1986)
69. Shanbhag, S., Schwan, N., Rimac, I., Varvello, M.: SoCCeR: Services over content-centric routing. In: Proceedings of the ACM SIGCOMM workshop on information-centric networking, ICN '11, pp. 62–67, ACM, New York, NY, USA (2011)
70. Chiocchetti, R., Perino, D., Carofiglio, G., Rossi, D., Rossini, G.: INFORM: a dynamic interest forwarding mechanism for information centric networking. In: Proceedings of the 3rd ACM SIGCOMM workshop on information-centric networking, ICN '13, pp. 9–14, ACM, New York, NY, USA (2013)

71. Qian, H., Ravindran, R., Wang, G.Q., Medhi, D.: Probability-based adaptive forwarding strategy in named data networking. In: 2013 IFIP/IEEE international symposium on integrated network management (IM 2013), pp. 1094–1101 (May 2013)
72. Lei, K., Yuan, J., Wang, J.: MDPF: an NDN probabilistic forwarding strategy based on maximizing deviation method. In: 2015 IEEE global communications conference (GLOBECOM), pp. 1–7 (Dec 2015)
73. Lei, K., Wang, J., Yuan, J.: An entropy-based probabilistic porwarding strategy in named data networking. In: 2015 IEEE international conference on communications (ICC), pp. 5665–5671 (June 2015)
74. Aloulou, N., Ayari, M., Zhani, M.F., Saidane, L., Pujolle, G.: Taxonomy and comparative study of NDN forwarding strategies. In: 2017 sixth international conference on communications and networking (ComNet), pp. 1–8 (March 2017)
75. Tsilopoulos, C., Xylomenos, G., Thomas, Y.: Reducing forwarding state in content-centric networks with semi-stateless forwarding. In: IEEE INFOCOM 2014—IEEE conference on computer communications, pp. 2067–2075 (April 2014)
76. Cui, Y., Lai, F., Yeh, E., Liu, R.: Enhanced VIP algorithms for forwarding, caching, and congestion control in named data networks. In: 2016 IEEE global communications conference (GLOBECOM), pp. 1–7 (Dec 2016)
77. Lai, F., Qiu, F., Bian, W., Cui, Y., Yeh, E.: Scaled VIP algorithms for joint dynamic forwarding and caching in named data networks. In: Proceedings of the 3rd ACM conference on information-centric networking, ACM-ICN '16, pp. 160–165, ACM, New York, NY, USA (2016)
78. Carofiglio, G., Mekinda, L., Muscariello, L.: FOCAL: forwarding and caching with latency awareness in information-centric networking. In: 2015 IEEE Globecom workshops (GC Wkshps), pp. 1–7 (Dec 2015)
79. Fu, B., Qian, L., Zhu, Y., Wang, L.: Reinforcement Learning-based algorithm for efficient and adaptive forwarding in named data networking. In: 2017 IEEE/CIC international conference on communications in China (ICCC), pp. 1–6 (Oct 2017)
80. Ioannou, A., Weber, S.: A survey of caching policies and forwarding mechanisms in information-centric networking. IEEE Commun. Surv. Tutor. **18**(4), 2847–2886 (2016). Fourthquarter
81. Zhang, Lixia, Afanasyev, Alexander, Burke, Jeffrey, Van Jacobson, claffy, kc, Crowley, Patrick, Papadopoulos, Christos, Wang, Lan, Zhang, Beichuan: Named data networking. SIGCOMM Comput. Commun. Rev. **44**(3), 66–73 (2014)
82. Afanasyev, A., Shi, J., Zhang, B., Zhang, L., Moiseenko, I., Yu, Y., Shang, W., Li, Y., Mastorakis, S., Huang, Y., Paul Abraham, J., Newberry, E., Liang, T., Schneider, K., DiBenedetto, S., Fan, C., Shannigrahi, S., Papadopoulos, C., Pesavento, D., Gordon, N.: NDN Delveloper's guide. Technical report, University of California, Los Angeles,The University of Arizona, Colorado State University, University of Pierre and Marie Curie, Sorbonne University, Beijing Institute of Technology, Washington University in St. Louis, The University of Memphis (2018)
83. Mahmudul Hoque, A.K.M., Amin, S.O., Alyyan, A., Zhang, B., Zhang, L., Wang, L.: NLSR: named-data link state routing protocol. In: Proceedings of the 3rd ACM SIGCOMM workshop on information-centric networking, ICN '13, pp. 15–20, ACM, New York, NY, USA (2013)
84. Lehman, V., Gawande, A., Zhang, B., Zhang, L., Aldecoa, R., Krioukov, D., Wang, L.: An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN. In: 2016 IEEE/ACM 24th international symposium on quality of service (IWQoS), pp. 1–10 (June 2016)
85. Kalghoum, A., Gammar, S.M., Saïdane, L.A.: Performance evaluation of interest traffic generation and forwarding strategy impact in ICN. In: 2016 IEEE/ACS 13th international conference of computer systems and applications (AICCSA), pp. 1–5 (Nov 2016)
86. Yi, Cheng, Afanasyev, Alexander, Wang, Lan, Zhang, Beichuan, Zhang, Lixia: Adaptive forwarding in named data networking. SIGCOMM Comput. Commun. Rev. **42**(3), 62–67 (2012)
87. Bastos, I.V., Moraes, I.M.: A forwarding strategy based on reinforcement learning for content-centric networking. In: 2016 7th International conference on the network of the future (NOF), pp. 1–5 (Nov 2016)
88. Posch, D., Rainer, B., Hellwagner, H.: SAF: Stochastic Adaptive Forwarding in named data networking. IEEE/ACM Trans. Netw. **25**(2), 1089–1102 (2017)
89. Yingming, W.: Using the method of maximizing deviation to make decision for multiindices. J. Syst. Eng. Electron. **8**(3), 21–26 (1997)

90. Li, C., Okamura, K., Liu, W.: Ant colony based forwarding method for content-centric networking. In: 2013 27th international conference on advanced information networking and applications workshops, pp. 306–311 (March 2013)
91. Eum, S., Nakauchi, K., Murata, M., Shoji, Y., Nishinaga, N.: Potential based routing as a secondary best-effort routing for information centric networking (ICN). Comput. Netw. **57**(16), 3154–3164 (2013)
92. Bastos, Ian Vilar, Moraes, Igor Monteiro: A diversity-based search-and-routing approach for named-data networking. Comput. Netw. **157**, 11–23 (2019)
93. Lee, Munyoung, Song, Junghwan, Cho, Kideok, Pack, Sangheon, Kwon, Ted Taekyoung, Kangasharju, Jussi, Choi, Yanghee: Content discovery for information-centric networking. Comput. Netw. **83**(C), 1–14 (2015)
94. Rao, Ravipudi Venkata: Decision making in the manufacturing environment: using graph theory and fuzzy multiple attribute decision making methods. Springer Science & Business Media, Berlin (2007)
95. Bastos, I.V., Moraes, I.M.: Diver: a diversity-based search-and-routing approach for named-data networking. In: 2016 IEEE global communications conference (GLOBECOM), pp. 1–7 (Dec 2016)
96. Ko, B.J., Pappas, V., Raghavendra, R., Song, Y., Dilmaghani, R.B., Lee, K., Verma, D.: An information-centric architecture for data center networks. In: Proceedings of the second edition of the ICN workshop on information-centric networking, ICN '12, pp. 79–84, ACM, New York, NY, USA (2012)
97. Chen, M., Weng, X., Wang, X., Xing, C., Zhang, G.: A mechanism of information-centric networking based on data centers. In: 2013 international conference on advanced cloud and big data, pp. 40–45 (Dec 2013)
98. Zhu, M., Li, D., Liu, Y., Wu, J.: CDRDN: content driven routing in datacenter network. In: 2014 23rd international conference on computer communication and networks (ICCCN), pp. 1–8 (Aug 2014)
99. Zhu, M., Li, D., Wang, F., Li, A., Ramakrishnan, K.K., Liu, Y., Wu, J., Zhu, N., Liu, X.: CCDN: content-centric data center networks. IEEE/ACM Trans. Netw. **24**(6), 3537–3550 (2016)
100. Xie, R., Wen, Y., Jia, X., Xie, H.: Supporting seamless virtual machine migration via named data networking in cloud data center. IEEE Trans. Parallel Distrib. Syst. **26**(12), 3485–3497 (2015)
101. Mansour, D., Tschudin, C.: Towards a monitoring protocol over information-centric networks. In: Proceedings of the 3rd ACM conference on information-centric networking, ACM-ICN '16, pp. 60–64, ACM, New York, NY, USA (2016)
102. Ireland, Kenneth, Rosen, Michael: Bernoulli Numbers, pp. 228–248. Springer, New York (1982)
103. Ibukiyama, Tomoyoshi, Kaneko, Masanobu: Generalized Bernoulli numbers, pp. 51–63. Springer, Tokyo (2014)

Dima Mansour is a Ph.D. candidate in the Computer Networks Group at the University of Basel, Switzerland. She received her master in Computer Science from Tishreen University, Syria. She is interested in the challenges and opportunities in supporting services over Named-Data Networking, from security and service management, to performance and load balancing.

Haidar Osman is a Tech Lead at Swisscom, Switzerland's leading telecommunication company. He acquired his Ph.D. in Software Engineering from the university of Bern in 2017. Besides his R&D activities in industry, Dr. Osman collaborates with the universities of Basel and Bern where he co-advises students and gives lectures in software quality, software engineering, and concurrent programming.

Christian Tschudin is a full professor at the University of Basel where he leads the Computer Networks Group. Before joining the University of Basel, he was a PostDoc at ICSI in Berkeley and associate professor at Uppsala University, Sweden. He earned his computer science Ph.D. from the University of Geneva and has a diploma degree in Mathematics. His research Interests are: Named-Function Networking, mobile code, artificial chemistries, wireless networks, and security.