Distributing Quality of Service (QoS) Policies in Name-based Networks

Ryo Yanagida, University of Glasgow
Jeremy Singer, University of Glasgow
Paul Harvey, University of Glasgow
Leon Wong, Rakuten Mobile, Inc.
Colin Perkins, University of Glasgow

***Abstract*—Much contemporary usage of the Internet is content-driven. The traffic originates from highly distributed services where users exclusively care about the data they are receiving, not where it comes from. However the longstanding architecture of the Internet and its management are largely host-oriented. Name-based networks address this dichotomy by proposing a content- and data-centric approach. In name-based networks, every interaction is driven by the name of the data that is being requested. Although this aligns with the large proportion of the way data is retrieved over the Internet, it remains an open question as to how this mechanism could be used to manage the forwarders in name-based networks and support autonomous operation. Specifically, in the context of a name-prefix based QoS mechanism, policies need to be present on every forwarder on the network, requiring policy distribution and continuous updates. Currently there is no standard solution to distribute and manage QoS policies on name-based networks. This paper explores how name-based networks could be managed autonomously, identifying mechanisms to control QoS policies across a network by examining an example scenario. We present a novel design for a QoS distribution mechanism that is conceptually coherent with existing name-based protocols; we subsequently illustrate this proposal by sketching out a concrete use case scenario.**

***Index Terms*—Information Centric Networking, Quality of Service, Autonomous Network Management**

I. Introduction

The Internet uses a host-based communication model that requires each party in a communication to know the location-based IP address of the other. This model was designed when the Internet was much smaller, when content often resided on a specific host, and connecting to that host was the only way to retrieve data. Today, however, most content is highly distributed across multiple sites, and the challenge is to retrieve it from the closest replica, improving latency and alleviating bottlenecks.

Name-based networking protocols, such as NDN [1] and CCNx [2], addresses this by changing the paradigm. Instead of a content consumer having to identify a host that has the data, name-based networks allow it to request content using its name as a first-class identifier. This permits features such as on-path caching and location transparency, which have the potential to help a wide range of applications, such as distributed ML [3], through improved efficiency in data transmission. The new paradigm also opens a pathway to enabling purely name-based Quality of Service (QoS) [4]. However, there are no standard mechanisms to configure and manage name-based QoS, making it hard to deploy.

In this paper, we explore some design considerations and approaches to distributing and updating name-based QoS policies, examine trade-offs and design choices, and discuss how named-based QoS policy distribution could be implemented for a residential ISP. Section II introduces Name-based Networking and the problem of distributing QoS policies. Section III, presents design considerations for a name-based QoS policy distribution mechanism. We outline an example scenario in Section IV, followed by discussion in Section V.

II. QoS Policies in Name-Based Networks

In Name-based networks, data is requested and retrieved by name rather than IP address. The consumer sends an Interest with the name, which is forwarded towards the producer using the name prefix, in a manner analogous to IP forwarding based on address prefix. The producer, upon receiving an Interest packet replies with the Data that follows the reverse Interest path back to the consumer.

QoS in name-based network is naturally implemented based on the name prefix of the data being requested [4], leveraging the semantics of names when choosing the packets for which QoS treatment is applied. The QoS policy, $\mathcal{P}_{n,t}$, applied at a node n at time t in such a system is naturally a set of mappings from name prefixes, P, to per-hop behaviours, B:

$$\mathcal{P}_{n,t} : P \rightarrow B \tag{1}$$

and the *QoS policy distribution problem* is the challenge of distributing $\mathcal{P}$ to the relevant forwarding nodes in a timely manner.

We illustrate this problem in the context of a residential Internet service provider (ISP) that has chosen to prioritise subset of traffic from video streaming services to their customers, and must distribute an appropriate QoS policy to the forwarders in their network. In an IP-based network, such forwarders might be configured by pushing an update to each device in turn. This is not possible in a pull-based name-based network, since the Data must be solicited by the Interest with the corresponding name in order for it to be forwarded. Forwarders must rather autonomously pull

the appropriate configuration by name, following the name-based paradigm. This leads to the following questions:

RQ1: What is the appropriate naming-structure for QoS policy distribution? An important question is how should hierarchical content names be structured in order to allow effective QoS prioritisation.

RQ2: How frequent are the QoS policy updates and what are the timing requirements for delivery? In a named-based system, where data is pulled by name, there is a concern that devices might need to poll for updates frequently, introducing overhead.

RQ3: What are appropriate mechanism to retrieve policy updates? The native name-based approach is to send an INTEREST packet to request the appropriate policy at an interval. However, this limits the timing of the update to the polling interval, affecting the timeliness of the policy distribution. We must consider alternative mechanisms to trigger the nodes' policy retrieval more dynamically.

In the following we explore these challenges, considering data naming and timing requirements for policy updates.

III. Design Consideration

In exploring how to distribute QoS policies in named-data networks, we first consider what elements are needed to construct a QoS policy distribution mechanism.

A. Naming QoS Policies

In name-based networking it is crucial to identify what must be named and how those names are structured. Considering QoS policy distribution, we identify four name components that makes up the structured name any QoS naming scheme must consider: **(i)** The **content name prefix** identifies the packets to which the QoS treatment must be applied, ensuring that appropriate per-hop behaviours are applied to INTEREST and/or DATA packets with names that are covered by the name prefix. **(ii)** The **QoS policy name** that is to be applied to content that matches the content name prefixes. This could be an exact match for a content name prefix, used to fetch the policy entry for the specific prefix, or a less specific content name prefix could be used to identify policies that apply to a range of content. **(iii)** The name of the **forwarder**, or group of forwarders, that apply the QoS policy. This might identify an individual forwarder or, identify a set of forwarders or a region. **(iv)** Finally, there may be a name indicating the **version** of the QoS policy to be retrieved.

These name components are combined to form structured names that identify either a single QoS policy or a set of QoS policies. By using the exact value in the respective name component, it could be used to refer to a specific QoS policy, while reserved names, such as `all`, `common`, or `latest`, can be used as a shorthand to refer to a group of policies or special instance of the policy.

Names in named-networking approaches are structured hierarchically. In the context of policy retrieval, one option is to position name components that define *where* a policy is to be applied within a network, such as those identifying the forwarder or group of forwarders, higher in the hierarchy and names relating to individual items that are subject to QoS policy application in the lower part of the hierarchy. That is, organise policy names from more general to more specific components.

Having forwarder components higher up in the hierarchy enables a shorthand to refer to all the policies with that particular set of forwarders, allowing the forwarder to fetch them all in one query. We refer to this type as **forwarder first** structure. This can be beneficial to provide granularity in policy application. A particular prefix could be dominating the traffic in an area, and to tame the load on the network a particular prefix within particular area might need to be de-prioritised.

It is also possible that a policy for a specific name prefix is required, in which case the forwarder and version could be optional and placed down in terms of hierarchy, and place the specific content name prefix in place of policy name. We refer to this as **content name first** structure.

Different QoS policy naming structures serve different purposes. The strict hierarchical structure of names in Name-based networking implies that aliases may be needed to cater for different needs.

B. Timing of Policy Updates

Considering the scenario outlined in Section II, there are three classes of updates to the policies that are needed:

Scheduled policy update is a type of update that occurs with known timing. This could be induced by contract changes with a service provider requesting the QoS policy treatment. For example a service provider may decide to add or remove a content name prefix, or group of prefixes, from a QoS policy at the time of establishing a contract or at renewal and/or cancellation. This could equally be caused by the users subscribing to a new service. In terms of name structure, use of reserved name 'all' in place of policy name component may be beneficial, as it allows the bulk fetch of a set of policies. In the place of version name component, use of names like 'latest' may be beneficial as a shorthand.

On-demand policy update is a type of update that occurs dynamically due to the traffic that passes through a node. This could be triggered by a pay-per-view service, for example permitting a customer to temporarily receive enhanced service when they subscribe to paid content. Another on-demand policy updates may come from the service providers requesting a QoS treatment on a new content with short notice. In such scenario, policy name could be queried higher up in the hierarchy. This allows much shorter name for fetching a specific policy for a content name prefix. Since update is requires spontaneously, querying for the content name as it appears is useful as it allows the forwarders on path to make use of the cached response from the controller.

Traffic management update is a type of update that occurs dynamically due to operator traffic engineering. Such updates can be scheduled unpredictably and the changes are often temporary, similar to on-demand policy updates based on user activity.

These classes of updates have further timeliness consideration in terms of distribution and application. Scheduled policy updates, as the name suggests, occur at a set timing, and the frequency of change is up-to how frequent the contract changes. Since the change is expected, timeliness of change is not dynamic, and therefore could be pre-distributed and scheduled for policy application. These can be fulfilled with a conventional pull mechanisms, where forwarders fetch their set of policies periodically.

The on-demand policy updates and traffic management updates can occur at unpredictable times. While there may occasions when such updates are *likely*, there are no set timings that can be pre-programmed. Since the utility of on-demand changes is short-lived, the timeliness of the update is important. This requires a more heavy handed update trigger to ensure a timely update. For example, RPC mechanisms may be used to trigger an update from controller side, and/or by utilising the content names to trigger updates.

C. Triggering QoS Policy Distribution

Since the pure name-based networking paradigm consists of Interest-Data interaction, QoS policy updates must be triggered by Interest packets sent by forwarders. The QoS policy distribution mechanism must consider how these updates are triggered.

Timer-based periodic fetch, when scheduled correctly, can achieve timely update as long as the timing is known. This is suitable for the initial setup of forwarders, and periodic updates. However, it will risk taking time to apply changes that are out of schedule and if the clock is out-of-sync, this will further delay the update. It is possible to pre-distribute the policy and schedule the policy application to reduce the chance of policy application being late.

For more dynamic scenario, there are two methods that can be chosen. **RPC triggered** updates can enable on-demand update triggered by the controller. RPC mechanisms such as RICE [5] and Reflexive forwarding [6] or even simpler Interest packet with parameters can be used to induce a policy fetch at a forwarder. However, this puts strain on the routing. The forwarder must have a prefix and a name for it to be routable, which adds to the routing table entry distributed in the network.

Traffic triggered on demand updates do not require any routing entries pointing to the forwarders with reduced timeliness in terms of updates. This approach will need a way to indicate the lifetime to enable forwarders to fetch updates and prevent policies to become stale.

D. Format of QoS Policy Updates

When forwarder requests a policy, or policies, triggered by one of the above mechanism and using one of the name structure described earlier in this section, there are four potential responses expected by the forwarder:

Policy entry: Individual prefix to forwarding behaviour mapping. Policy name portion of the name structure could be replaced with the content name prefix.

Policy index: Reference to policy in the form of a list of names that can be used to fetch the policies. Policy name portion of the name structure could be set to a reserved name such as `index`.

Policy set: Bundled set of policy (all or subset). Reserved name such as `all` or `common` can be used in policy name or forwarder portion of the name structure.

Policy set index: Reference to a set of policies.

The policy entry and set contains the key-value pair(s) holding the content name-prefix and the respective QoS policy. Indexes hold reference(s) to the respective set or entry and are useful to discern whether the forwarder has the set of policies or not. In the cases for a one-off or known schedule updates, policy sets can be fetched to prevent having to retrieve each policy individually.

IV. Example

Section III described different components and aspects of the name-based QoS distribution mechanism. This section describes how these could be combined in an example scenario, considering the factors described in the Section II.

We consider the scenario described in Figure 1, where a residential ISP agrees to cooperate with a set of video streaming services providers to prioritise a subset of their traffic. In addition, the ISP also offers premium QoS service and uses QoS policies to manage traffic in their network.

A. Naming of QoS Policies

The QoS policies for Service A in Figure 1 can use a single prefix, since the content prefixes that requires QoS treatment are readily identified. For example, the name `/isp_a/conf/qos/policy/common/service_a/live/` is a possible name for the QoS policy used by the `live/` content name prefix of `service_a`. We can also include the entry in a bundle used by all the forwarders, for example: `/isp_a/conf/qos/policy/common/all` might be used to name all of the common policies across the network; while `/isp_a/conf/qos/policy/common/service_a/all` might name all of the policies for `/service_a/` prefix.

Services requiring more dynamic policies pose a challenge, however. For example, service B in Figure 1 offers a premium service to select users. One way of enabling this in name-based manner is offering a temporary prefixes. This poses two challenges: one is updating QoS policy in the controller dynamically, and the other challenge is distributing QoS policy for the correct path dynamically. The former needs coordination with the service provider to update the policy in the controller. The latter requires forwarders to fetch them dynamically. Since the location of the customer is not known, one approach is to let these prefixes reside within the `common` forwarder group,

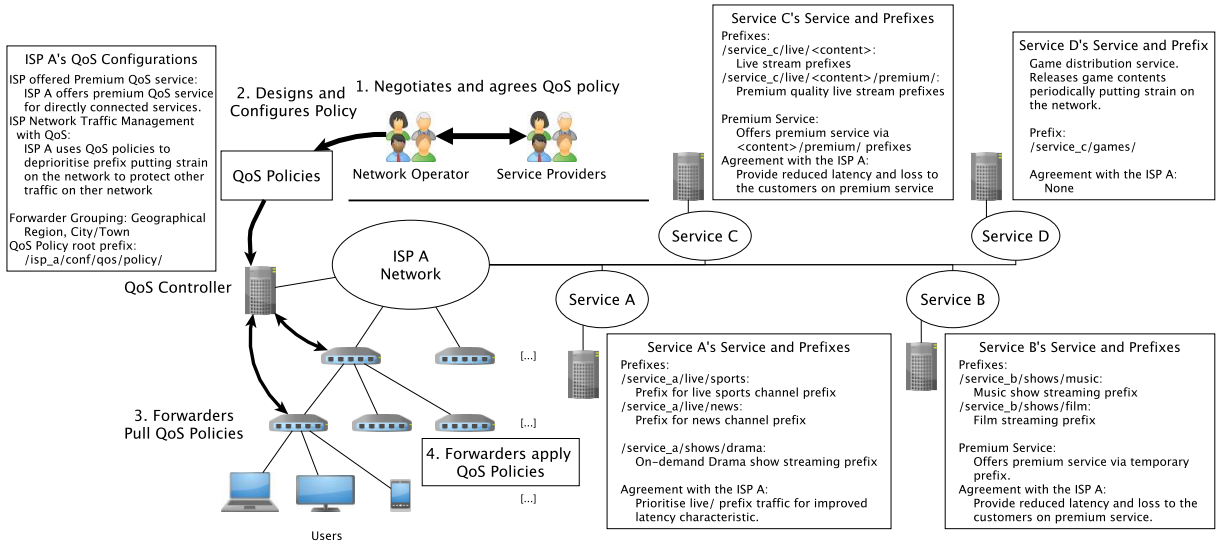

Figure 1: Diagram showing example QoS scenario.

increasing rate of policy set updates, adding to the overhead for the scheduled update across the network, and bloating the policy tables in the Forwarders. One way to avoid this would be to use an additional namespace in forwarder name component, e.g., `on_demand/`, such that the any forwarder that receives those on-demand traffic can fetch them regardless of the geographical grouping. e.g., `/isp_a/conf/qos/policy/on_demand/service_b/<tmp name>`

As mentioned in Section II, the content name structure is crucial. Another approach to enabling dynamic QoS policy application might be the approach used in Service C, Figure 1, where `premium/` name component exists under the content name. This, however, leads to bloat in terms of the number of policy entries required since content is selected by prefix, requiring each `<content name>/premium/` prefix to be distributed and occupying a large number of entries for QoS policies.

If the ISP offers premium QoS to customers for the streaming services, it will need to determine the path to each such customer by examining the network topology and communicate this to the QoS controller. This then needs to be added to the respective forwarders, under forwarder name component of the QoS policy name structure. Suppose the host is connected via Forwarders with IDs A1 and B2. The following two prefixes could be used to name a policy that prioritises traffic to Service A's `shows/` contents that were not initially requested by the service provider but offered by the ISP as an additional service: `/isp_a/conf/qos/policy/forwarder-a1/service_a/shows/` and `/isp_a/conf/qos/policy/forwarder-b2/service_a/shows/`

In cases where the ISP aims to de-prioritise particular traffic where the network exhibits high load, to protect other traffic, depending on the locality, new policies would be added under the forwarder name component identifying the region where these forwarders may reside. Suppose forwarders in region A1 are experiencing a sudden load on service D distributing games under `games/` prefix, the following prefix could be used to name a policy deprioritising the traffic: `/isp_a/conf/qos/policy/region-a1/service_d/games/`

B. Timing and Mechanisms for Policy Updates

Updates and timing requirements expected for Service A, in Figure 1, are straight forward as they demand a network-wide treatment agreed for a set duration. Changes to the set could be distributed using the **timer** based approach described in Section III-C where the policies are fetched in advance or on schedule.

More dynamic scenarios similarly pose challenges in name distribution requirement, timing requirement, and mechanisms to cater them. For example, In the case of Service B in Figure 1, the new entry is required frequently and in a timely manner since end users expect a paid service to be available immediately. Therefore, the forwarders are expected to detect the priority service in use and fetch the policy immediately. In this case, any INTEREST packet for `/service_b/` prefix received by the forwarder will need to trigger an on-demand policy fetch, as the QoS controller will have no way of knowing which forwarders to trigger the QoS policy updates.

For Service C in Figure 1, the frequency of policy updates at the QoS controller will be less, and will require fewer entries than Service B, but will require more entries compared to Service A. Therefore, the Forwarders can either take the overhead and hold the larger amount of policies in advance, or reduce the amount by dynamically fetching as the Interest for `/service_c` prefix is received. On the other hand, ISP offered premium QoS treatment will

likely to be a similar frequency to Service A, whereby users choose the premium option in a known period. Therefore this can be enabled with **timer** based update.

For a network traffic management scenario where the operator aims to deprioritise some traffic that is putting strain on the network, the frequency or the timing is not predictable, and the timeliness is important. Since the prefix is not known in advance, the controller must trigger updates to the relevant forwarders with an **RPC** based mechanism. This suggests that such management traffic will also need to be subject to the QoS prioritisation to retain control of the nodes.

V. Discussion

Section IV described how the approaches in Section III can be combined to enable QoS operations in the context of residential ISP network with the presence of other services. From this example scenario, we can observe obstacles and challenges in enabling name-based QoS, but more importantly challenges in configuring and managing configurations dynamically in the name-based networks.

A. Naming

One aspect that is apparent from the example scenario in Section IV is the importance of a naming structure, both in terms of content name structure offered by the data producer as well as policy name structure that the network operator uses.

Service C, in Figure 1, keeps content that requires different QoS policies together under the same content name prefix. This prevents aggregation of the names in terms of the level of service offered and bloats the name-prefixes needed to enable QoS in name-prefix approach as shown in Section IV-A. However, there may be a legitimate reason from the service provider's perspective to approach the naming in such a way. For example, one reason to keep the different quality of the same content under the same content name prefix is the locality of the data. Having the content name prefix before the quality of the content will prevent duplicate prefixes. If the quality of the content is earlier than its name, then there will be one under `/[...]/standard/<content name>/` prefix and `/[...]/premium/<content name>/` prefix. And these two may refer to the same topological location. As name prefix is used to route packets topologically towards the data producer, the efficiency of name-prefix routing table is affected by the naming structure. By keeping the standard and premium content under the same `/[...]/<content name>/` prefix, the data could reside in one topological location without requiring an additional entry each additional quality level.

This shows the conflict between the two different perspectives in the use of naming. The structured hierarchical naming scheme is fundamental to grouping of names using prefixes. When two different uses of a name require distinct grouping, there lies a conflict in which one use case is catered while the other faces an overhead. Aliasing is an approach that may potentially help alleviate this, enabling aliasing is non-trivial [7], and one of the prominent name-based protocol NDN does not support aliasing [8]. While the content producer can change the name structure, this leads to changes in application and while it may be feasible to retain the legacy namespace, this presents further indirection to be maintained within the content producer, and leads to overheads, such as duplicate Pending Interest Table entries and content storage (on-path caching) entries.

We presented several ways a QoS policy and a set of policies could be named. The most basic approach to naming the QoS policy for a given content name prefix would be to use the said content name prefix as part of the name, however, requesting every policy using these names pose a large overhead in terms of the RTT and sheer number of Interest-Data packet exchange. One approach to reduce the RTT overhead is to bundle them into a set, whereby one Data packet under one name contains multiple policies. While this reduces the number of requests, it requires a mapping from the raw policy names to the aggregate set name to be defined.

Another aspect that must be discussed is the naming of forwarders and their grouping. To apply QoS policies with more granularity in terms of the forwarder and where they are topologically, this information needs to be encoded into the name, allowing the forwarders to fetch the correct policies.

B. Timing of Policy Updates

From the example scenario, in Figure 1, we observed different timing requirements. This includes timing for changes, distribution, and application. To distribute a policy in name-based networks, the forwarders must request an update via an Interest packet. While increasing the frequency of periodic update can improve the timeliness of policy distribution and application, this causes a large overhead in terms of packet exchange. We proposed two solutions to alleviate this issue in Section III-C: (i) use of RPC to forwarders and (ii) traffic-induced policy fetch. RPC requires a new set of routing table entries to route the traffic to the forwarders, and latter introduces latency between the time of content name-prefix discovery from the Interest packets and receiving the policy for the said content name-prefix. While this work focuses on QoS policy distribution, timeliness of the configuration distribution and application can be an important factor in a wider network configuration and management context; the timing concerns of policy updates likely apply to those wider contexts as well.

C. Mechanism for QoS Policy Distribution

Name-based networking paradigm decouples the end hosts in the process of data transmission, which brought the benefits of name-based approach, such as interest aggregation, on-path caching, and visibility to the metadata

of the data being carried by the network through its name-prefix. However it comes with side effects. One of which is the inability to identify the end-to-end data transmission between the two specific hosts. This poses challenges in enabling a more granular QoS policy application. In an IP network, identifying the connection can be done with a tuple containing source and destination hosts' respective IP address and port number. Whereas in name-based networks, INTEREST packets are aggregated, and there is no source and destination address pairs nor port numbers to identify any 'connection'. This motivates approach of Service B in Figure 1, for example, where each user is given a temporal prefix to request their data. This however is at odds with some of the benefits of name-based networks, such as INTEREST packet aggregation and on-path caching, and poses challenge in terms of timely fetch of the respective policy.

A name-based approach requires any data to be pulled by the host. This poses challenge in enabling a timely update of the forwarder's QoS policies as shown in Section IV, requiring extra round trip to trigger a pull from the forwarder. In fact, this applies to any other configuration that needs to be distributed to the forwarder.

D. Limitations and Future Work

This work has not examined the interaction between different policies whereby multiple policies conflict with each other. Similarly to earlier work on name-based QoS [4], this work does not examine caching, both in terms of caching DATA packets that holds the policy nor in terms of QoS policy for caching strategies. This work explored video transmission use-case, but our approach could benefit other ICN applications e.g. [3]. In addition, our approach may also integrate to next-gen cellular networks as well [9]. Such topics remain for future work.

VI. Related Work

Ambalavanan et al. [10] also explore the design space of QoS mechanisms for NDN, although at a higher level of abstraction than our work. They favour a client-driven RSVP-style protocol, although they highlight scalability concerns due to the storage of per-flow state in the network.

Gundogan et al. [11] describe an experimental NDN testbed deployment of QoS for an IoT scale system. Their QoS control covers holistic resource allocation including forwarding and caching. However like most prior research in this area, they explicitly state that they need more research to 'securely administer and distribute QoS settings at runtime', which is the basis of our proposed contributions.

Bari et al. [12] introduce a novel QoS policy management and enforcement scheme that leverages the facilities of software-defined networking (SDN) to enable dynamic adaptation of control plane rules based on changing traffic patterns. Our name-based approach enables similar programmatic configuration of policy, although with a more declarative approach.

VII. Conclusions

We explored how name-based QoS policies can be distributed across the network in a name-based manner, following on from previous work on name-based QoS mechanism [4]. We proposed design considerations in terms of naming requirements, timing requirements, and mechanisms needed to distribute QoS policies with the example scenario. The findings in this work will inform our future work, such as implementation and evaluation of a QoS policy distribution mechanism in a name-based networking protocol, and exploring caching QoS policies and their policy distribution mechanism.

Acknowledgements

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript arising from this submission.

This work was funded in part by Rakuten Mobile, Inc.

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM CCR*, vol. 44, no. 3, Jul. 2014.

[2] M. Mosko, I. Solis, and C. A. Wood, "Content-Centric Networking (CCNx) Semantics," RFC 8569, Jul. 2019.

[3] W. Geng, Y. Zhang, D. Kutscher, A. Kumar, S. Tarkoma, and P. Hui, "SoK: Distributed Computing in ICN," in *Proceedings of the 10th ACM Conference on Information-Centric Networking*, ser. ACM ICN '23. New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 88–100.

[4] R. Yanagida, J. Singer, P. Harvey, L. Wong, and C. Perkins, "Name-based quality for name-based networks," in *45th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2025.

[5] M. Król, K. Habak, D. Oran, D. Kutscher, and I. Psaras, "RICE: Remote method invocation in ICN," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ser. ICN '18. New York, NY, USA: Association for Computing Machinery, Sep. 2018, pp. 1–11.

[6] D. R. Oran, D. Kutscher, and H. Asaeda, "Reflexive Forwarding for CCNx and NDN Protocols," draft-irtf-icnrg-reflexive-forwarding-00, Oct. 2024.

[7] E. Jung, "A design of alias naming scheme for namespace in named data networking," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Jan. 2016, pp. 537–540.

[8] O. Karrakchou, N. Samaan, and A. Karmouch, "ENDN: An Enhanced NDN Architecture with a P4-programmabIe Data Plane," in *Proceedings of the 7th ACM Conference on Information-Centric Networking*. Virtual Event Canada: ACM, Sep. 2020, pp. 1–11.

[9] P. Suthar, M. Stolic, A. Jangam, D. Trossen, and R. Ravindran, "Experimental Scenarios of Information-Centric Networking (ICN) Integration in 4G Mobile Networks," RFC 9269, Aug. 2022.

[10] U. Ambalavanan, N. Nayak, D. Grewe, and N. Mohan, "Resource reservation in information centric networking," in *Proceedings of the 9th ACM Conference on Information-Centric Networking*, 2022, p. 165–167.

[11] C. Gündoğan, J. Pfender, M. Frey, T. C. Schmidt, F. Shzu-Juraschek, and M. Wählisch, "Gain more for less: The surprising benefits of QoS management in constrained NDN networks," in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, p. 141–152.

[12] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: An autonomic QoS policy enforcement framework for software defined networks," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.