Grant Agreement No.: 644663
Call: H2020-ICT-2014-1
Topic: ICT-05-2014
Type of action: RIA

architectuRe for an Internet For Everybody

D2.2: Initial Component and Interface Specifications

Work package	WP2
Task	T2.2
Due date	31/10/2015
Submission date	31/10/2015
Deliverable lead	InterDigital Europe
Version	1.0
Authors	Teemu Kärkkäinen (TUM), Charalambos Theodorou (Avanti), Arjuna Sathiaseelan (UCAM), Joerg Ott (TUM)
Reviewers	Alicia Higa, M. Potts

Abstract	This document describes initial components and interface specifications.
Keywords	Information-centric networking, IP-over-ICN, Fronthaul, Edge caching

Disclaimer

The information, documentation and figures available in this deliverable are written by the RIFE project consortium under EC grant agreement 644663 and do not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

Copyright notice

Acknowledgment

This report is partially funded under the EC H2020 project RIFE, grant agreement 644663.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	R[1]	
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to bodies determined by the RIFE project	
CO	Confidential to RIFE project and Commission Services	

[1] R: report, P: prototype, D: demonstrator, O: other

EXECUTIVE SUMMARY

This document presents the initial component and interface specification of the architectural framework that the RIFE project defines, utilizing advances in Information-Centric Networking (ICN) to enable local users to access IP-based as well as novel ICN-based services over a local operator network.

In this context, we review the baseline ICN architecture that has been achieved within the PURSUIT project since this architecture lays the foundations for our design work. This baseline ICN architecture is presented in terms of its core functions, namely rendezvous, topology management and forwarding.

Based on this, we describe the design methodology of the ICN architecture in RIFE, focusing on the specification of key components and interfaces as we foresee within this framework as well as the integration of delay-tolerant networking (DTN) solutions and satellite backhaul technology into the overall system architecture. In addition, the design of the local network's edge cache with its interfaces and functional models is described.

This is an evolving document that defines an initial component and interfaces as we expect within the RIFE architectural framework. It is not a complete architectural description and can involve constant update throughout the remainder of the project.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AS	Autonomous System
BF	Bloom Filter
BGP	Border Gateway Protocol
BGW	Border Gateway
BRP	Bordercast Resolution Protocol
CoAP	Constrained Applications Protocol
DAGS	Directed Acyclic Graphs
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
DTN	Delay-tolerant networking
FId	Forwarding Identifier
FN	Forwarding Node
HTTP	Hypertext Transfer Protocol
ICN	Information-Centric Networking
IP	Internet Protocol
LId	Link Identifier
NAP	Network Attachment Point
NAT	Network Address Translation
RV	Rendezvous
SDN	Software-defined Networking
TCP	Transmission Control Protocol
TM	Topology Manager
UE	User Equipment

1 INTRODUCTION

The RIFE project is chartered with developing solutions that bring affordable Internet connectivity to those who can currently not afford such services. For this reason, RIFE will employ advances in Future Internet (FI) technologies, such as Information-Centric Networking (ICN) or Delay-Tolerant Networking (DTN), to assemble a system architecture that will allow for keeping costs of deployment at an affordable level for communities, while providing state-of-the-art as well as novel community-oriented services.

RIFE achieves the objective of affordable Internet provisioning through three key innovations. Firstly, RIFE will localize communication through the concept of **edge caching**. This will push existing Content Delivery Network (CDN) concepts further to the edge of the network, lowering costs for backhaul connectivity as one major cost factor in Internet deployment. Secondly, RIFE will enhance the efficiency of such edge caching by **contextualising** its usage in the sense that special focus is given on content that is relevant to the local community. With this, edge caches will not only serve the short tail of the content space (within the typical Zipf distribution of content popularity) but also the content relevant to the community which is placed along the long tail, i.e., the space of content usage which is usually not served by current CDN systems. Lastly, RIFE will provide innovative solutions to **reduce fronthaul costs** through improved operational efficiency, while enabling novel services along the spectrum of health, Internet of Things, educations and other crucial areas.

Keeping these innovation areas in mind, D2.1 Usage Scenarios & Requirements has set out to outline use case where such innovations come to full fruition, while also providing Key Performance Indicators (KPIs) to evaluate the efficacy of the solutions developed. D2.2 Initial Component and Interface Specifications provide the second step by outlining the system architecture in which our solutions will be developed as well as the components within such architecture. We will also outline the main component-level interfaces, albeit at a high level due to the early stage of our work – the future D2.3 Final Component and Interface Specifications will provide a more detailed and stable interface description, capturing the insights from our work until then.

This deliverables is structures as follows. In Section 2, we start with a brief insight into ICN, which forms the basis for our system architecture. We then present our system architecture in Section 3, including the integration with DTN as well as the integration with work done in the POINT project[2] on providing IP-based services over ICN, both aspects directly influencing the system architecture. Section 4 focuses on the interface description, as outlined in the system architecture in Section 4. Due to its central role in our innovation-driven approach, as outlined above, Section 5 provides first insights into the design of an edge cache that enables the localization of popular as well conceptually relevant content at the border of the fronthaul network. Our conclusions are presented in Section 6.

[2] http://www.point-h2020.eu

2 ICN BACKGROUND ARCHITECTURE

In the RIFE ICN architecture, we build on an existing set of key design tenets for an Information-centric Network that will ultimately enable the realization of our key innovations. This enhanced architecture in RIFE will be deployed based on a previous design solution proposed and developed in the PURSUIT project [Tro2011]. In this section, we provide a brief background on the PURSUIT ICN architecture. The reader is referred to available literature on ICN in general and the PURSUIT solutions in particular, such as described in [Tro2011][Tro2012} for more information on this subject.

2.1 Design Tenets

At the heart of this ICN architecture is a set of design tenets that form the basis for a functional view of an information-centric architecture by outlining aspects of the manipulation of information flows, together with building transient communication relations between computational entities, as outlined in [Tro2012].

The **first** such tenet is that of *providing means for identifying individual information items*. Such identification can be realized, for instance, through some form of naming scheme or through flat labels. Specifically, PURSUIT utilizes statistically unique fixed-size labels. Such labels hold no explicit semantics and have no direct meaning to most networking components. However, relations to various naming schemes can be realised through algorithmic associations between an information identifier and the designated names, such as through hashing a name into the used label identifier [Fot2010].

In PURSUIT, the information identification places individual information items into a context; we call this *scoping* as the **second** tenet. Each information item is placed in at least one scope. Scopes are sets of information therefore, they are information themselves. Hence, they are identified as such through the same fixed-size label mechanism and can be nested in another scope. Combining the first and second tenet allows for building complex directed acyclic graphs (DAGs) of information over which computation can take place [Tro2012].

The **third** tenet exposes *a service model* that directly operates on the information graph. With that, any design can implement computational problem solutions through computation over an information flow among entities that utilize the exposed service model. Within PURSUIT, a publish/subscribe service model is realized. In such publish/subscribe service model, information items are published by those who are willing to offer them (i.e., Publishers), and subscribed to by those who are interested in them (i.e., Subscribers). This model is utilized for all forms of communication, including those among the networking functions described below. It is worth noting that maintaining publications and subscriptions does not require any form of synchronization, which allows publishers and subscribers to not only be spatially decoupled, but also temporally. Furthermore, because relations are established on the basis of information rather than location endpoints, publishers and subscribers do not need to be aware of each other's locations. When there exists a mutual interest in an information item, a publish/subscribe relation is established, which may ultimately result in the transmission of data from the information publisher to the subscriber [Tro2012]. Lastly, it is important to note that the Pub/Sub model of the underlying ICN architecture can easily implement other communication semantics, such as send and receive (or request and response).

The **fourth** tenet outlines the *functional model* of the underlying network architecture that realizes the manipulation of information through the exposed service model. These core functions consist of (i) rendezvous (RV), topology management (TM), and forwarding (FN).

The RV (rendezvous) function matches the demand for and supply of information, resulting in the creation of a Pub/Sub relationship at any point in time. When a publisher wishes to provide information, it publishes its availability to the RV function, which will position it within the namespace defined for the given root scope, therefore adding the publisher to the set of publishers of this information item. Interested subscriber(s) wishing to receive this information may send subscription request(s) to the RV function [Tro2012]. In many instances albeit not necessarily, the realization of the RV function requires maintaining a state space that is aligned with the namespace as defined by the application or the overall system. When a match occurs, the RV function initiates the creation of an appropriate communication relation between the matched publisher(s) and subscriber(s) by publishing a topology formation request to the TM function. In a sense, the RV function acts as an agent that links the pub and sub through the underlying supply-demand information and sends a new topology request to the TM.

The TM (topology management) function allows for forming a suitable communication relation between given publisher(s) and subscriber(s). When the TM receives a topology formation notification from the RV, it selects the appropriate network resources to be used according to the optimisation objective of the routing algorithm in operation. In link-based environments, for instance, various forwarding mechanisms such as line-speed publish/subscribe internetworking (LIPSIN) [Jok2009], multi-stage Bloom filters (MSBF) [Tap2012] or Bloom filter switching [Tsi2013] can be used to represent a suitable communication relationship. In this case, the TM will create a Forwarding Identifier (FId), which is provided to the publisher in order to specify the packets' route in the network as a source route.

The final core function, i.e., the FN (forwarding) function, delivers information from the source(s) to the sink(s), i.e., the publisher(s) to the subscriber(s). This operation is aligned with the TM function in that the forwarding information determined in the TM function is used in the forwarding operation.

The **fifth** tenet defines the concept of *dissemination strategies*, which (according to [Tro2012]) *"… defines the particular implementation for the aforementioned core functions but also the aspects regarding information space governance and management."* In that sense, a dissemination strategy limits a specific realization of the core functions to a particular sub-space of the information space. In other words, the notion of dissemination strategies decides how information is propagated in parts of the overall information space. Examples for such strategies are link-local, domain-local and global strategies. [Tro2012] outlines how the concept of dissemination strategies defines the layering that takes place in the resulting system through encapsulating solutions to problems, realized in one strategy and therefore with its specific realization of the core functions, provides other solutions to larger problems. With this, recursive layering is at the heart of the architecture.

2.2 Platforms

The feasibility of this approach is demonstrated in a networked environment via the realization of the information management, the exposed service model, as well as the particular node implementation. For implementing the first two tenets, the information structure supported by the PURSUIT prototype [Tro2012] is that of a directed acyclic graph (DAG). It is this DAG structure that provides a simplistic form that can capture many

existing application concepts (e.g., ontological concepts, complex event processing, etc.), promising therefore an easy mapping of these higher-level concepts onto the abstractions provided by the PURSUIT architecture. The information structure is maintained according to the dissemination strategy that is assigned to each topmost scope.

While this architecture does not mandate any particular forwarding mechanism, one significant forwarding mechanism used by the platform in [Tro2012] is LIPSIN [Jok2009], which defines each directed edge in the network with a fixed-size Bloom Filter (BF)-based Link Identifier (LId). Using LIPSIN as the forwarding mechanism, each forwarding node in the network accomplishes a simple and fast packet processing by performing bitwise AND/COMPARE logical operations to test the membership of the node's Link Identifier (LId) in the FId of any received packets. The solution in [Tap2012] provides a version of BF-based forwarding that extends LIPSIN in terms of scalability and supporting true false-positive-free forwarding operations.

2.2.1 Use Within RIFE

Within RIFE, we utilize the aforementioned concept of dissemination strategies by realizing each IP protocol abstraction mapping as a separate dissemination strategy. This is driven by the fact that each protocol, e.g., at the IP level or the HTTP level, provides its own semantics in terms of communication and naming semantics, which maps well onto the separate dissemination strategies in PURSUIT. Hence, each protocol abstraction establishes its own namespace with its own root scope, while the operations of the core functions (RV, TM and FW) slightly differ in order to accommodate the semantics needed.

In practice, this use of dissemination strategy to provide a particular behaviour is very powerful and provides for an efficient implementation, while being able to re-use the platform being built and tested within the PURSUIT project. For example, rather than defining new protocol headers, or rigidly dividing the address space, as might be required in traditional layered protocols, the dissemination strategy is implicitly derived from the root scope of the information space or the protocol semantics that it represents (e.g., the response in the HTTP mapping is realized with a different dissemination strategy compared to that used for the request). We will outline this integration with the underlying ICN architecture in the various places of the RIFE solution.

3 RIFE PLATFORM ARCHITECTURE

This section outlines the system architecture of RIFE, starting with the main drivers for its design, leading to the chosen approach for our design. Due to the impact on our system design, we also outline the integration with DTN and the integration with IPoverICN solutions developed in the POINT project.

3.1 Main Drivers

The RIFE proposition is aiming at a single operator, improving its overall IP service offerings towards its customers. In order to make such proposition attainable, the following characteristics of the propositions are desired:

- To not change existing UEs, as far as possible (for emerging systems, such as IoT devices, where standardisation is still evolving, the platform should offer access to its native ICN functions).
- To modify as little as possible the UE software.

Furthermore, it is expected that our proposition will benefit from operator-internal SDN upgrades, i.e., piggybacking on the successful uptake of SDN in bringing about software-controlled, flow-managed, networks. In addition, our proposition provides a longer-term future proof migration through native ICN options that are enabled by our solution. Both of these trends, however, are left out from this deliverable, but they will be developed in future deliverables.

3.2 A Gateway-based System Architecture

Figure 1 below outlines our approach to the system architecture. In order to preserve the IP interfaces towards the UEs and its applications, RIFE uses a *gateway approach*, where the bridging between IP and ICN is performed in the Network Attachment Points (NAPs), i.e., the access gateways from customers to the network, and the ICN border gateway (ICN BGW), i.e., the access from and to peering networks or operator server resources.

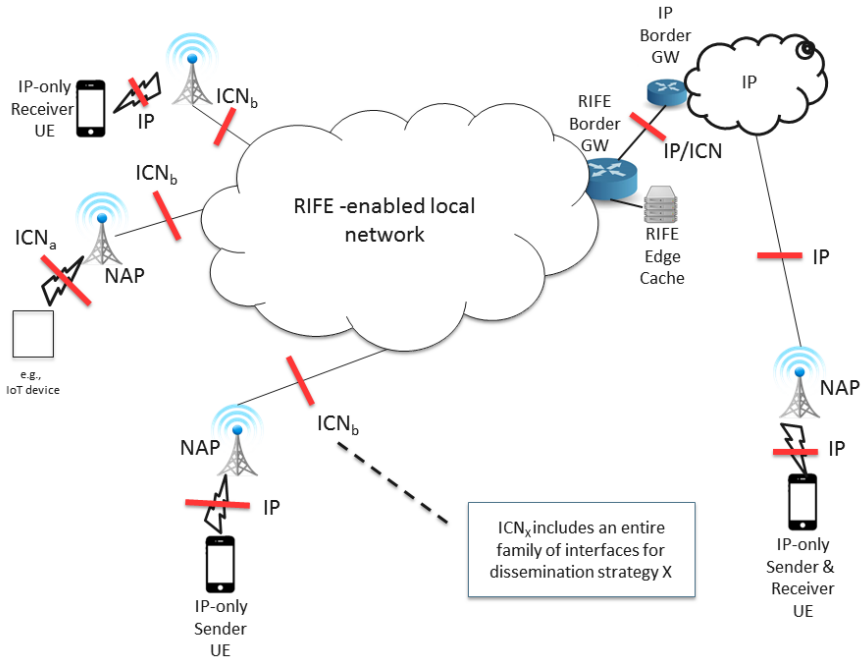


Figure 1 The RIFE system architecture

This approach allows for focussing on improvements to the fronthaul network through QoS and caching solutions as well as the support for DTN operations and services, while utilizing the innovations provided by existing IP-based services in the Internet. Furthermore, this choice of system architecture integrates well with the solutions developed in the POINT effort, utilizing the innovations developed in these efforts for the purposes and use cases of RIFE, while enriching them with our novel edge caching and DTN solutions.

In our architecture, the NAP serves as an IP-ICN gateway, handling all the offered abstractions supported by the IP interface (e.g., IP, TCP, HTTP, CoAP), while all communications beyond the ICN BGW complies with standard IP technologies, making the ISP network appear like a standard IP autonomous system. Furthermore, the NAP provides standard gateway functions such as Network Address Translation (NAT), firewalling and dynamic IP address assignment wherever required. In addition, in an effort to lower peering traffic to other IP networks, the ICN GW provides edge caching capabilities (shown as a separate logical entity in Figure 1) that not only transparently capture cacheable content but also enables proactive content placement for reducing any future requests to the content. Section 5 presents more details regarding the design for this component.

From our overview in Figure 1, we derive the more detailed interface view of Figure 2, showing the network-internal interfaces in addition to the interfaces facing the UEs and peering networks.

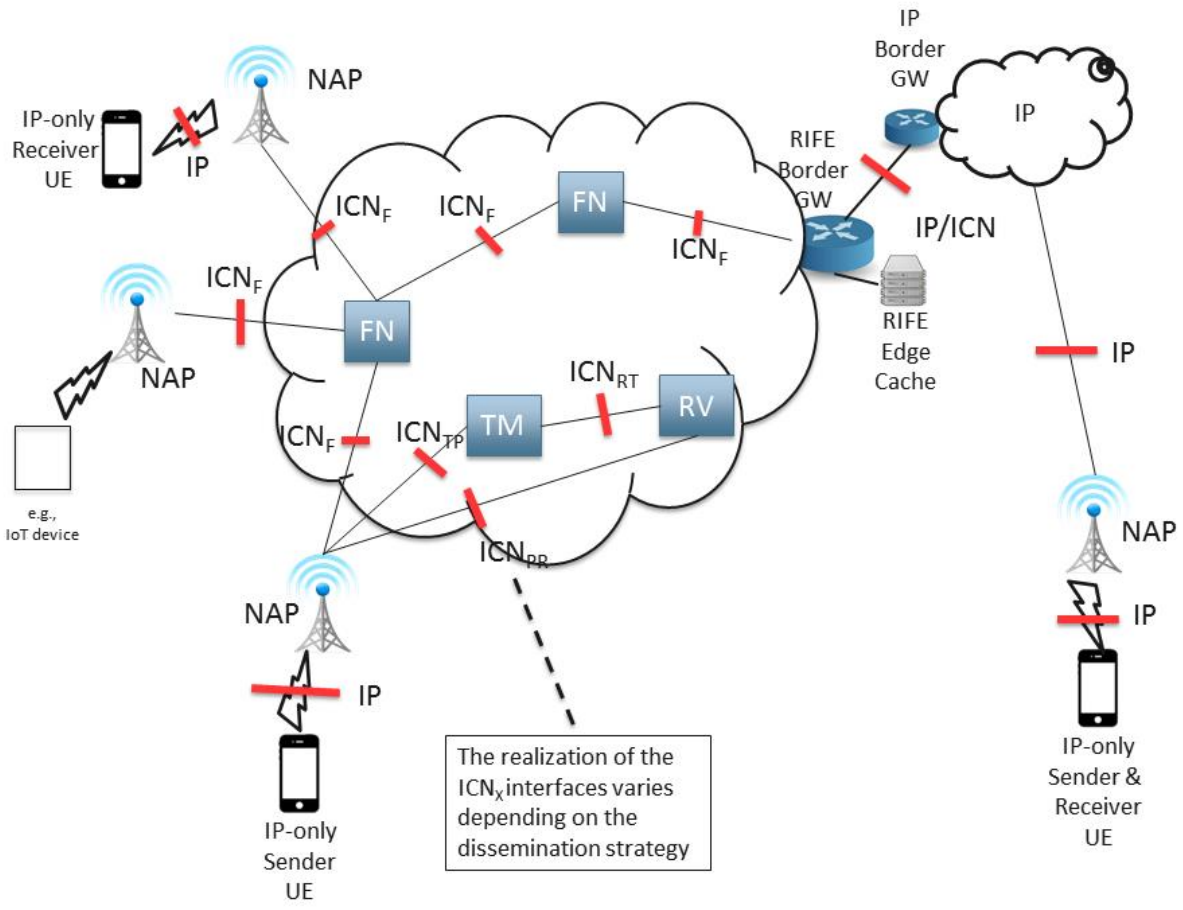


Figure 2 The RIFE system with network-internal interfaces

Within the operator network, the main ICN core functions are realized, i.e., RV, TM and FN, as described in Section 2. The diagram shows one RV and one TM, although in

practice a number of them may be used to achieve scalability and to reduce ICN matching latency [Alz2015]. The FW function is realised through forwarding nodes (FNs) that are placed at key operator switching points, in the same manner as existing routers or switches.

3.3 IP-over-ICN Mapping

The gateway-based system architecture in Figure 1 allows for not only exposing native ICN functionality to novel services but it also enables the provisioning of IP-based services over the ICN-enabled transport network of the operator. This aspect is important in order to enable migration from IP-based services to ICN-based ones but also to capture that it is still happening, and will likely continue to happen for some time, in the IP-based Internet as we know it today.

For preserving this ability to provide IP-based services to RIFE users, we integrate the solutions being developed in the POINT project. These solutions map IP as well as HTTP protocols at the NAP and the RIFE border gateway (see Figure 1) onto ICN in an efficient and scalable manner, enabling the re-introduction of multicast into HTTP-based services, the flexible provisioning of surrogate servers and other benefits (see [Tro2015] for more information on the opportunities of IP-over-ICN solutions).

The basic idea behind the IP-over-ICN is to map an *IP address* of any device to an appropriate *ICN name*, while the communication over this name follows a channel semantic, i.e., publishers will send packets over this name after the initial match has been made. The receiver of an IP packet, i.e., the entity that is meant to receive data on that IP address, subscribes to the appropriate ICN name for its own IP address, while any sender publishes to the appropriate ICN name.

The design of the specific namespace and the operations on this namespace are described in more detail within the POINT project [Poi2015]. Due to the similarities of the system architecture between RIFE and POINT, both following a gateway-based approach, it is the NAP of Figure 1 that realizes the IP-ICN protocol mapping. However, we also want to be able to send to ANY IP device, not only the ones connected to the ICN network. For this, a similar mapping is performed at the ICN border GW, with more details to be found in [Poi2015].

For deployment scenarios in RIFE, we assume in many cases an operator ICN network that serves a single IP namespace, which is managed via DHCP at each NAP. More complex IP address space cases, i.e., the existence of several independent namespaces across a single operator, can be captured through the prefix-based solution developed in POINT, also accommodating the possibility to host several gateways to peering networks within the operator network.

The efforts in RIFE in this space, focuses on transferring the solutions of POINT into the platform being used in RIFE, with both projects utilizing the same basic ICN platform. The SDN integration achieved in POINT, however, is not utilized in RIFE so that the forwarding function fully utilizes the core ICN function of the existing PURSUIT prototype. For more information and details on the specific solution, the reader is referred to [Poi2015] and the POINT deliverables in general.

3.4 Integration with DTN

The design tenets laid out in Section 2.1 that define the basis of the information centric architecture, together with the gateway-based RIFE system architecture, allow for the flexibility needed to provide a range of realizations suitable for different use cases and

scenarios. An important aspect of this flexibility is the fifth design tenet, *dissemination strategies*, which refers to the different specific connectivity options underlying the common information-centric (publish/subscribe) abstraction, as shown in Figure 3.

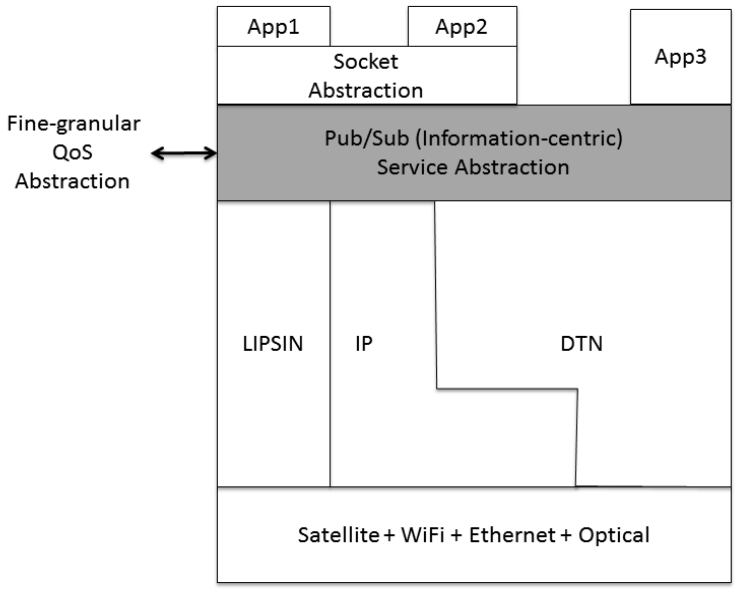


Figure 3 The RIFE platform with ICN and DTN integration

While a dissemination strategy within a local graph (e.g., LIPSIN) is likely to be the primary connectivity realization in many real-world deployments, there are scenarios where the nodes are never organized in a fully-connected graph due to, for example, lack of a stable infrastructure. In such cases, a delay and disruption tolerant dissemination strategy based on store-carry-forward networking can be used. For this purpose, the RIFE platform integrates Delay-Tolerant Networking (DTN) [RFC4838] as a fronthaul dissemination strategy, implemented using the Scampi [Kar2012] middleware, which is based on the Bundle Protocol [RFC5050].

Conceptually the DTN architecture defines named endpoints and forwarding of self-contained bundles between them through send/receive semantics. This is contrary to the first two ICN design tenet that call for named information, rather than named endpoints, and the general philosophy that *what* is being communicated is more important than *who* is communicating. However, the Scampi platform layers publish/subscribe semantics on top of the DTN architecture, resulting in a close match with the ICN architecture.

The DTN semantics, as refined by Scampi, can be mapped to the design tenets as follows:

- Tenet #1 (identifying individual information items): Scampi defines semantically meaningful, self-contained messages as the fundamental forwarding unit. These can be mapped directly to information items in the ICN architecture. Each has a unique identifier derived from the content, and are assigned with an arbitrary, application dependent service identifier when being published. The service identifier can be mapped to ICN information identifiers.
- Tenet #2 (information scoping): The service identifier described previously can be used as a structured information scoping mechanism similarly to ICN.
- Tenet #3 (service model): Scampi provides a publish/subscribe service model that matches the ICN semantics.

- Tenet #4 (functional model): In general, how the functions of the ICN architecture (*rendezvous*, *topology management*, and *forwarding*) map to Scampi functionality are dependent on the exact configuration of the instance and can be realized through a number of ways. The *forwarding* function is always implemented by exploiting short pair-wise contacts. When two nodes meet, some subset of the information items are exchanged depending on the routing algorithms and policies employed. The *topology management* is principally implemented by the routing algorithms and forwarding policies. These can range from completely random (epidemic routing) to exact, pre-computed routes over the space-time contact graph. The *rendezvous* function can be completely local, in which case the applications will only interact with the local information store. Alternatively, the publish/subscribe knowledge can be used in the routing process, or a search mechanism can be used to connect information publishers and seekers.

In terms of the RIFE system architecture, multiple integration approaches are possible. The simplest approach is to treat the DTN and ICN fronthauls as fully separate. In this approach, the RIFE gateway will decide to use one or more of the fronthauls based on policy, application, content or other criteria. However, since the DTN and ICN elements may co-exist in the same nodes within the fronthaul, it is possible to build an interface between them to provide "breakout/break-in" between the two dissemination strategies. Breakout from ICN to DTN allows dissemination to areas of poor connectivity (e.g., in case the ICN infrastructure is damaged) and a break-in from DTN to ICN can provide more efficient and quick transfer of the messages over the available ICN infrastructure. Finally, it is possible to provide a DTN-to-ICN conversion in the NAP, similarly to the IP-to-ICN conversion.

Beyond the systems level integration, it is possible to build high level application integration. Here the integration point could be in the caching system. For example, in the Web for RIFE implementation, a Squid cache storing Websites could be used to interact with Web/Scampi and Web/ICN implementations, with both networks remaining otherwise separate.

4 INTERFACE SPECIFICATIONS

This section defines the interfaces appropriate for the realization within our architecture framework, aimed at providing specification guidance for the initial realization.

4.1 UE IP Interface

In order to allow NAP support of all upper layer protocols via standard IP stack, the UE IP interface is defined as the communication interface between a UE and a NAP. This allows any end device running its packet exchange over IP to communicate with the NAP in a seamless manner, by utilising the standard IPv4 [IET1981] / IPv6 [IET1998] stack. Based on the underlying technology (e.g., IEEE 802.3 or 3GPP PDCP), IP packet is encapsulated into Layer 2. This ensures the NAP is able to support natively all upper layer protocols utilising the IP stack, e.g., DNS, DHCP, HTTP, ICMP, SMTP or IMAP.

4.2 RIFE Border GW IP Interface

In one deployment option, the RIFE border interface remains local to the RIFE deployment and it is implemented physically at the interface between the local RIFE network and the traditional broadband VSAT terminal. This solution is based on the existing Avanti infrastructure and the satellite terminal interfaced with the RIFE border gateway enabling the connection of the end users of RIFE to traditional IP networks.

For this, the RIFE border GW exposes an IP interface to the backhaul infrastructure, which will be implemented via a traditional VSAT broadband terminal, collocated with the ICN counterpart at the local network premises, as illustrated in Figure 4.

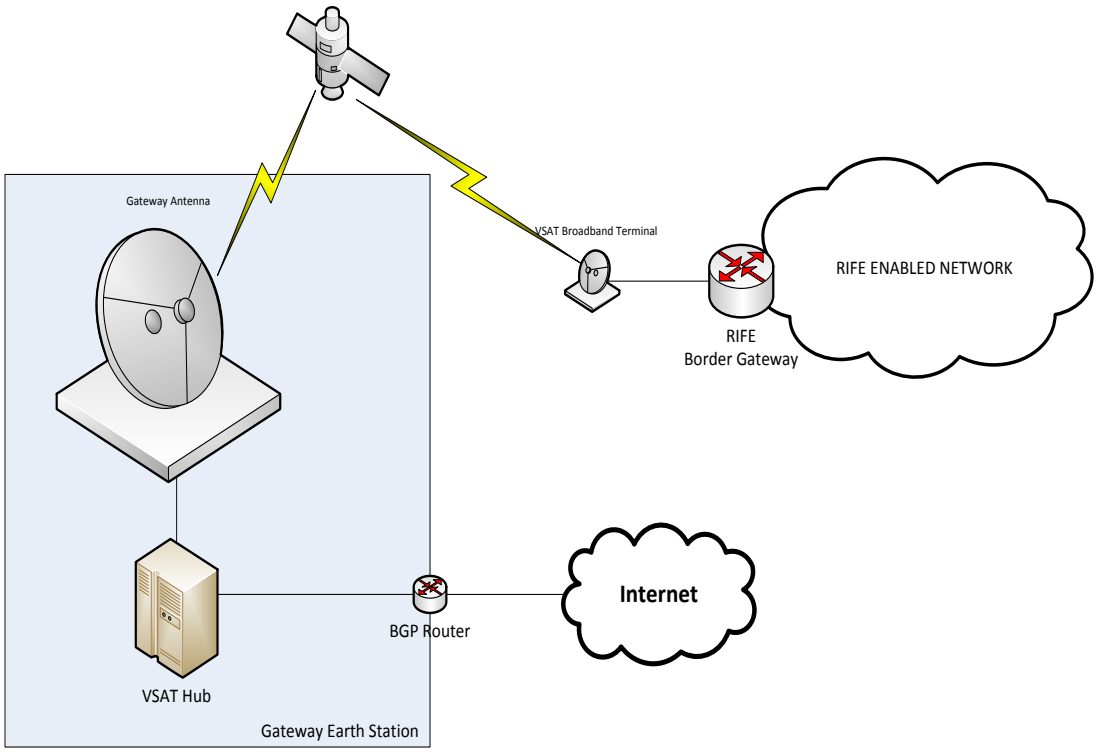


Figure 4 RIFE Border GW IP Interface

The terminal will provide seamless IP connectivity peering to standard IP networks via the Internet and present itself as a LAN interface to the ICN GW. The VSAT satellite network system provides a variety of standard and specialized IP features designed to minimize

space segment latencies and support standard and advanced IP networking protocols and services.

The network layer IP features provided in the system include:

- Bandwidth conservation features:
  - IP header compression
  - Payload compression
  - PEP protocol implementation
- IP packet delivery prioritization
- NAT/PAT
- Port Mapping
- VLAN tagging

The Application layer IP features in the system include:

- DHCP service/DHCP relay
- DNS caching
- IP Routing Protocols (RIPv2, BGP)
- Virtual router redundancy protocol (VRRP)

Specifically for IP peering to standard IP networks over the internet, the VSAT interface offers support to BGP. This feature provides a BGP-aware path between a remote network (Autonomous System in BGP terms) and a standard IP network (AS in BGP terms). Using BGP, the VSAT and the IP gateway operate as BGP edge routers and exchange routes with their BGP peers. The routing protocol between the VSAT and IP satellite gateway at the other end is still kept as BRP (Bordercast Resolution Protocol) although BRP has been enhanced to allow carrying certain BGP attributes such as the AS list.

4.3 RIFE Border GW ICN Interface

In the second deployment option, shown in Figure 5, RIFE includes the satellite backhaul as a physical ICN over satellite implementation. This allows positioning of the RIFE border gateway at the Gateway Earth Station after the satellite link and next to the border of the IP networks. As this implementation requires further development on the satellite link including satellite modems, it is considered to be out of the scope for the RIFE project. This kind of ICN over satellite implementation will be considered in future project work.

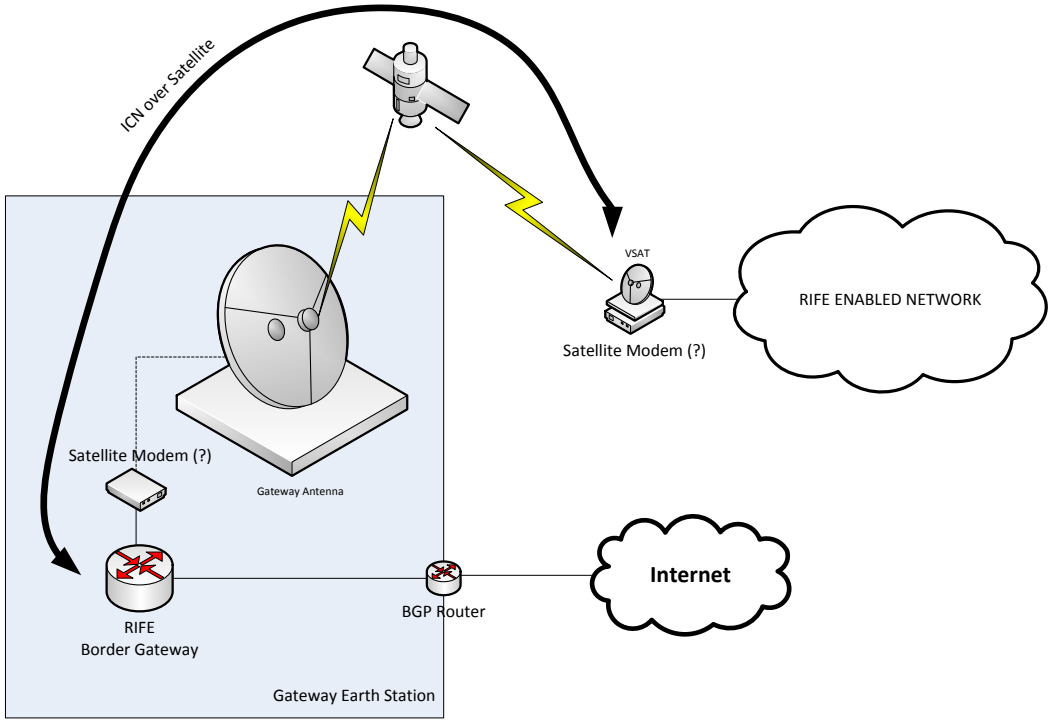


Figure 5 RIFE Border GW ICN Interface

4.4 Network-Internal ICN Interfaces

In RIFE, we define a number of network-internal ICN interfaces that realize the functionality of transporting IP services as well as native ICN services over the RIFE network, these interfaces are based on the work in [Tro2011][Tro2012].

4.4.1 Rendezvous-Topology Interface (ICN-RT)

The communication interface between the rendezvous and topology manager is defined as Rendezvous-Topology Interface (ICN-RT). It follows our ICN Pub/Sub communication mode by utilizing a predefined scope. This allows delivering topology formation requests from the RV to the topology manager. A topology formation request from the RV to the TM in our architecture requires:

- **Scope Path:** The scope path is defined as the path between the information items provided by a Publisher and requested by a Subscriber. The scope path is maintained by RV and must always be maintained in its full length and can consist of multiple fragments. Published information is given a scope path by the RV and this scope path shall be made available to the TM when subscribers submit a request for that item.
- **Information Identifier:** This identifier is responsible for the matching between a publication and a subscription when required. The relevant information identifier is to be made available to the TM from the RV when subscribers submit a request for that item.
- **Publishers:** Additional information the RV is required to inform the TM about is the list of publishers that advertise the information. When the publication is a request for information, only the identifier of the publisher that caused the corresponding match in the rendezvous is provided to the topology manager. This will always be one publisher in the case of unicast IP.

- **Subscribers:** Likewise, the RV is required to inform the TM of the list of subscribers that registered interest in the information to the RV. This will always be one subscriber in the case of unicast IP.

4.4.2 Publisher-Rendezvous Interface (ICN-PR)

The communication interface between a NAP and the RV is defined as the Publisher-Rendezvous Interface (ICN-PR), for advertising the interest of the NAP in publishing a particular information item. Whenever the NAP receives an incoming packet from a UE over its IP interface, this packet becomes an information item which needs to be published under a scope path, using explicitly or implicitly agreed namespaces. To allow RV to process the information item advertisement, the NAP shall provide the following information about the desired publication:

- **Scope Path:** The NAP shall inform the RV under which scope path the information item should be published and it must always be provided in its full length and can consist of multiple fragments.
- **Information Identifier:** The NAP shall inform the RV about the identifier of the information item. It should be published under the scope path provided.
- **Strategy:** The dissemination strategy of this request shall be made available to RV from the NAP, in most cases this strategy is defined through a standard root scope in the scope path.

4.4.3 Subscriber-Rendezvous Interface (ICN-SR)

In order to express the NAP's interest in obtaining information published under a particular information item, the Subscriber-Rendezvous Interface (ICN-SR) defines the communication interface between a NAP and the RV function. It must indicate this to the RV by subscribing to a particular information item under a scope path, using explicitly or implicitly agreed namespaces, so that the NAP can receive any information advertised by a publisher. To allow RV to process the subscription request, the following information is to be received from NAP:

- **Scope Path:** In which scope path the desired information item will be published (and therefore where this subscription expects a successful match). This must always be provided in its full length and can consist of multiple fragments.
- **Information Identifier:** The identifier of the information item under which the NAP shall receive data.
- **Strategy:** The dissemination strategy under which the information is published, this is assumed to be defined by the subscriber through the root scope in the scope path.

The ICN-SR interface allows the NAP to indicate its interest in all information items published under a scope path without specifying the exact information item using explicitly or implicitly agreed namespaces.

4.4.4 Topology Manager-Publisher Interface (ICN-TP)

The Topology Manager-Publisher Interface (ICN-TP) interface defines the interaction between the TM and a publisher. It has been selected by the TM to provide the data for a matched publication and subscription. Once a positive match occurs between a publication and a subscription, the TM creates the delivery FId, and delivers the FId to the

corresponding publisher, and finally instructing the latter to start forwarding data through the network. This instruction involves the TM providing the publisher with the following information:

- **Scope Path:** In which scope path the desired information item will be published, this must always be provided in its full length and can consist of multiple fragments.
- **Information Identifier:** The relevant information identifier, for which a match between a publication and a subscription has occurred, published under the Scope Path.
- **Strategy:** The dissemination strategy under which the information should be published, this is assumed to be defined through the root scope in the scope path.
- **FId:** The forwarding identifier relevant to the dissemination strategy used for the forwarding and to use this identifier to start forwarding data.

Transmission of the relevant data through the ICN-F interface will be invoked once the publisher receives the Fid.

4.4.5 Forwarding Interface (ICN-F)

The interaction between a publisher that needs to place data on the first forwarding node (FN) and the interaction between FNs in the network is defined as the Forwarding Interface (ICN-F). Upon receiving instructions via the ICN-TP interface, the publisher provides the FN entity of its corresponding node with encapsulated data that include the following:

- **Payload:** The payload data is required to be forwarded in a suitably encapsulated packet, which contains the following information.
- **FId:** The FId is used by the publisher to pass to it by the TM in the packet header.
- **Scope Path:** The scope path under where the desired information item has been published. The scope path must always be provided in its full length and can consist of multiple fragments.
- **Information Identifier:** The information identifier with which the item was published.

It is the first FN who identifies the FId in the packet header, which is then compared against its forwarding information to decide where to send the packet next. At the last FN aimed at the subscriber, the Packet Payload and Scope Path is decapsulated and passed to the subscriber, who will then take the desired action depending on the Information Identifier, as represented by the Scope Identifier(s) and the Information Item Identifier.

5 EDGE CACHE DESIGN AND INTERFACES

Our system architecture, as shown in Figure 1, enables the localization of communication through providing content locally as much as possible. For this, the RIFE border GW is extended through an edge cache, currently provided as an HTTP-level cache. Such placement of an HTTP-level proxy server provides seamless external access to Web content. A proxy server stands in between a number of clients and outside remote servers, which makes efficient content caching one of the most important features of proxy servers especial when they are located at the edge of the network, i.e. edge cache. The edge cache captures incoming traffic and decides whether the content shall be cached or forwarded without caching, while replying to incoming requests with already cache content. With that, RIFE places popular contents closer to the end users. Therefore, users near edge cache servers can enjoy much shorter average response times and better quality of experience, while the operational costs of the backhaul connectivity (see Figure 1) is reduced. This makes it more attractive for a group of people with the same interests who are using the same proxy server.

Due to the important role of edge caching in localizing communication and therefore reducing overall costs, we will describe in the following aspects of the design and implementation on how caching at the edge server is realized. We define a policy-based edge caching approach to manipulate content at edge servers, allowing for users to place content proactively into the edge cache, while the edge cache can retrieve such proactively placed content at off-peak times, thus further reducing costs of delivery. For this, the edge cache shall be capable of computing the most cost-efficient methodology to place the content over distributed edge server nodes, allocating policy-based cache resources and queuing methods to store contents (whole page or fragments) based on predefined caching rules and thereafter allowing restricted and authenticated access to the subscriber group. Hence, our approach is geared towards cost-efficient cache management (based on content to be published) and will be implemented using publically available open source Web proxy solutions over a European-wide testbed that can offer customized access control with policy-based cache separation. At this stage of our work, we utilize the openly available Squid[3] proxy and extend the platform according to the following description.

5.1 Process of Proactively Placing Content

Our initial goal for the edge cache, apart from providing standard proxy-based caching, is the ability to proactively place content into the edge cache, which in turn can be retrieved according to defined policies (e.g., retrieval at off-peak times). We foresee the following process to realize such proactive content placement:

1. Interaction with the end user via a Website-based form entry, allowing the user to submit the policy request.
2. Realize a policy agent that does the following:
    a. Create a directory with quotas;
    b. Temporarily configure Squid to use this directory only;

[3] http://www.squid-cache.org/

c. Issue HTTP requests to the set of URLs u (which Squid will now proxy upon return in the chosen directory);

d. Re-configure Squid to use now again all directories, i.e., the previously configured ones as well as the newly created one.

3. During the content fetching phase, the cache selection algorithm (e.g., RR) will NOT use the regular cache directory. Squid will, however, continue to use ALL directories for retrieving cached content (determined through the digest).
4. After the content fetching phase, the proactively placed cache directory is added to the Squid cache directories and will therefore be used for future requests.
5. A background daemon run that removes a directory from the exclusion list after its retention time t expires, therefore adding the directory to the normal pool of directories that Squid uses.

5.2 Components and Interfaces

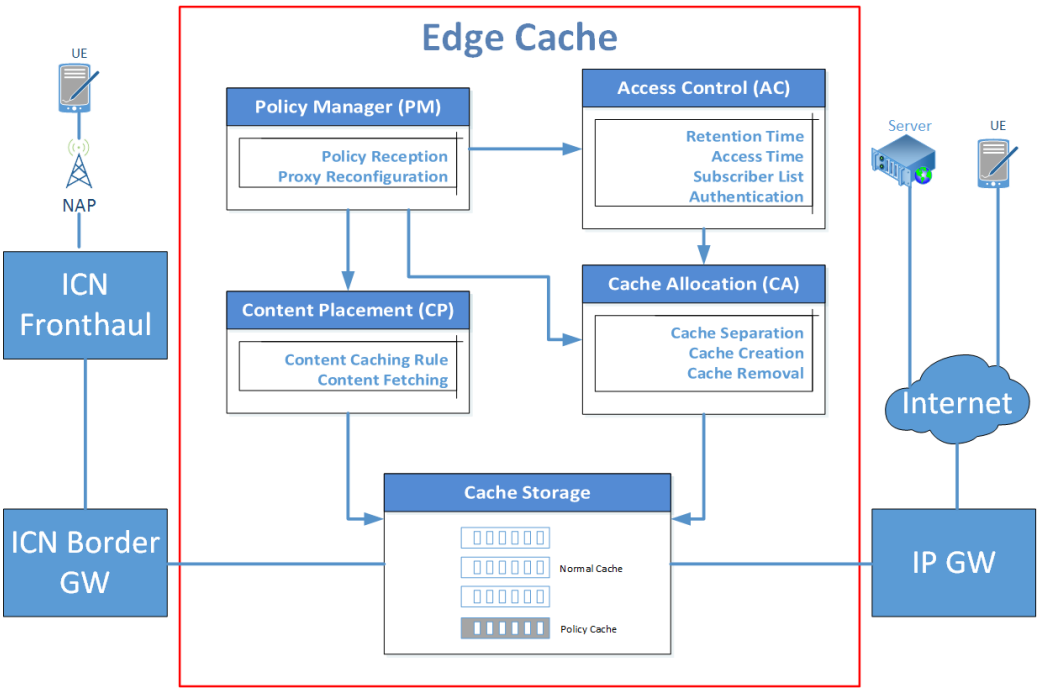


Figure 6 Initial component and interface design of edge cache

Figure 6 illustrates the main components and interfaces of our edge cache platform. Within our overall system architecture of Figure 1, our edge cache resides in-between the ICN and IP interface of the RIFE border GW. For this, we have split the border GW in Figure 6 into the ICN and IP part (ICN GW and IP GW) to illustrate the position of the RIFE edge cache. With this, any outgoing traffic from the RIFE ICN-enabled fronthaul will be routed first through the edge cache before being relayed over the IP GW. With the interfaces of the RIFE Border GW in mind, as discussed in Sections 4.2 and 4.3, respectively, the physical location of the edge cache might be at the RIFE-enabled fronthaul network premises or in the satellite control centre (in the case of positioning the IP GW in the satellite control centre). In the latter case, we will assume the inclusion of ICN capabilities into the edge cache, something that remains to be studied in future work. For now, we assume the placement of the edge cache at the fronthaul network premises for cases of IP termination at the RIFE border GW.

With this in mind, we defined the following components in the edge cache:

1. ***Policy Manager (PM)*** defines the policy for proactively placing content in the edge cache. It performs two key tasks as computational problems, namely (i) *Policy Reception:* upon receiving a new policy request from publisher that includes requested cache size, URL list, retention time, and client list, it updates the proxy configuration parameter stored as local variable, and (ii) *Proxy Reconfiguration:* based on the new configuration parameter as updated by the policy reception, this function reconfigures the proxy with new access control methods based on the client list.
2. ***Content Placement (CP)*** based on content caching and new policy rule, fetch the contents and place it to the dedicated cache:
    a. *Content Caching Rule:* This rule defines what contents is to be cached, this can be based on URL or file type, and for example, we can define this rule so that all contents associated with a URL are to be cached.
    b. *Content Fetching:* Based on the URL and content caching rule, we use a simple http get request to fetch the requested content to the dedicated server.
3. ***Cache Allocation(CA)*** performs optimal allocation policy that is implemented via two stage of allocation, 1) in-node multi-cache allocation based on policy-based caching requirements, 2) inter-node cache allocation aimed at distributing the cache capacity across the network under storage budget constraints (optional, this can work with content placement where resource is limited in the network). This module is the manager of the physical ***cache storage*** and is responsible for:
    a. *Cache Separation:* The type of cache shall be pre-assigned based on the content characteristics, the content can utilize different types of storage spaces:
        a. In-memory cache: Provides faster file retrieval, but the size of a cache is limited.
        b. Disk cache: Allows storing a larger amount of cacheable objects with the trade-off of a less effective response time.
    b. *Cache Creation:* Based on the requested cache size, this function creates new dedicated cache for storing the content requested for the specific publisher.
    c. *Cache Removal:* Based on the retention time, any dedicated cache prescribed by publisher over the expiry date will be removed subject to capacity constraints.
4. ***Access Control(AC)*** defines how the content can be accessed based on:
    a. Allowed user lists
    b. Retention period
    c. Allowed access time
    d. Authentication methods (if applicable)

Figure 7 shows a possible platform deployment of the components in Figure 6 through a separate VM, specifically within a server in the RIFE overlay testbed.

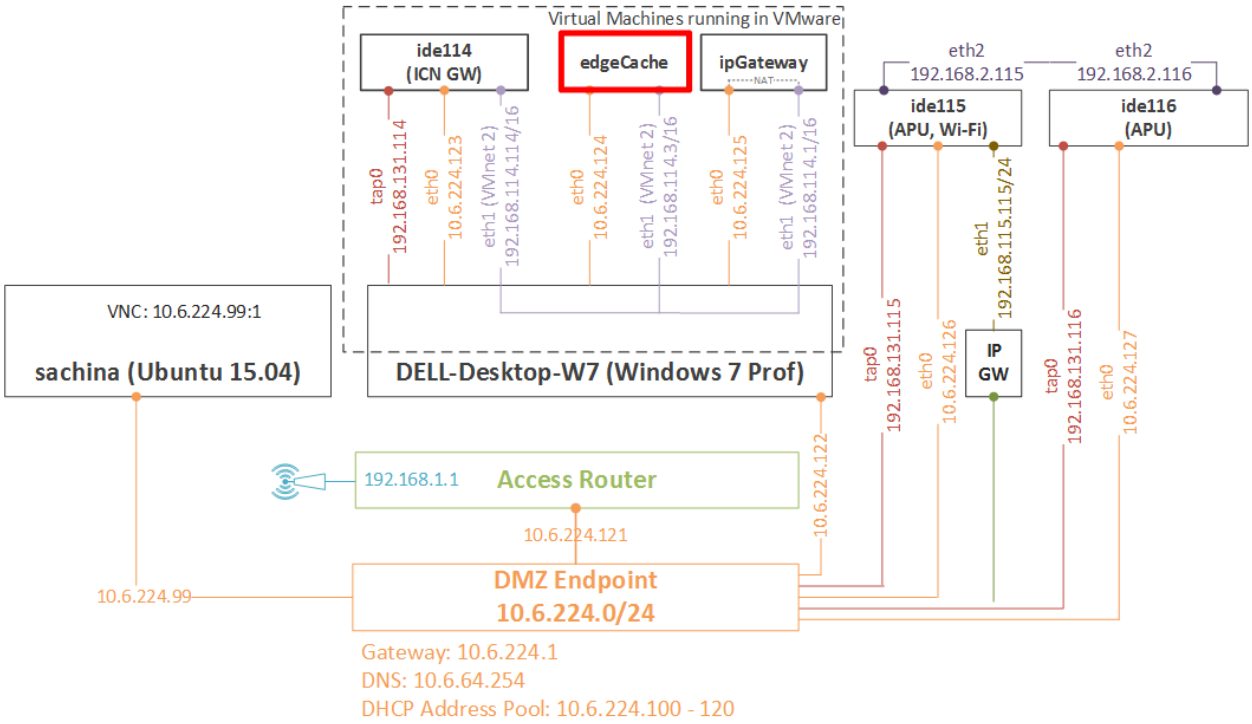
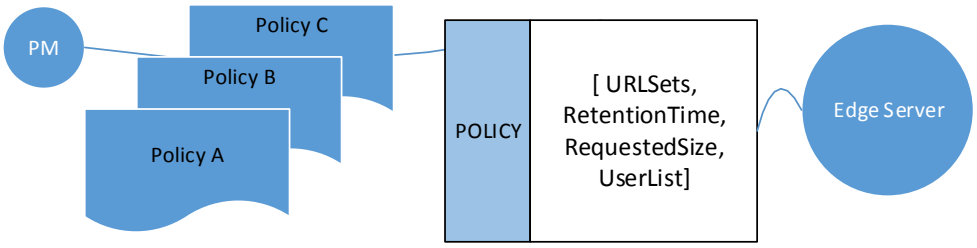


Figure 7 Testbed component and interface reference diagram

5.3 Solution realization

The process, outlined in Section 5.1 involves main steps as shown in Figure 8:

1 PolicyMaker sends the request:[URLSets, rentionTime, requestedSize, userList]



2 Apply squid_policy_open.conf

Lock Current Cache	cache_dir ufs /var/spool/squid1 1 100 16 256 no-store cache_dir ufs /var/spool/squid2 1 100 16 256 no-store cache_dir ufs /var/spool/squid3 1 100 16 256 no-store
Create New Cache	cache_dir ufs /var/spool/squidNew_1 1 **PolicyRequestedSize** 16 256
Allow dst URL	acl policy_url dstdomain **PolicyURLSets** http_access allow policy_url
Deny all other IP	http_access deny all_other_ip
Apply Conf	Sq-reconfig && sq-flush (squid -z) (Here it will create the missing Cache)

3 SquidClient sends HTTP request to fetch the URL sets from network source

Fetching contents	Squidclient **PolicyURLSets**

4 Apply squid_policy_close.conf

Lock Policy Cache	cache_dir ufs /var/spool/squidPolicy 100 16 256 no-store
Apply access control	Http_access allow **PolicyUserList**

5 Remove cache when retention time expires

Cache Removal	Background bash script

Figure 8 Procedures for realization of the policy-based caching mechanism.

The implementation of the above procedures involves the interaction amongst different functional models between the client and server as Figure 9 shows the interaction of the related information modules.

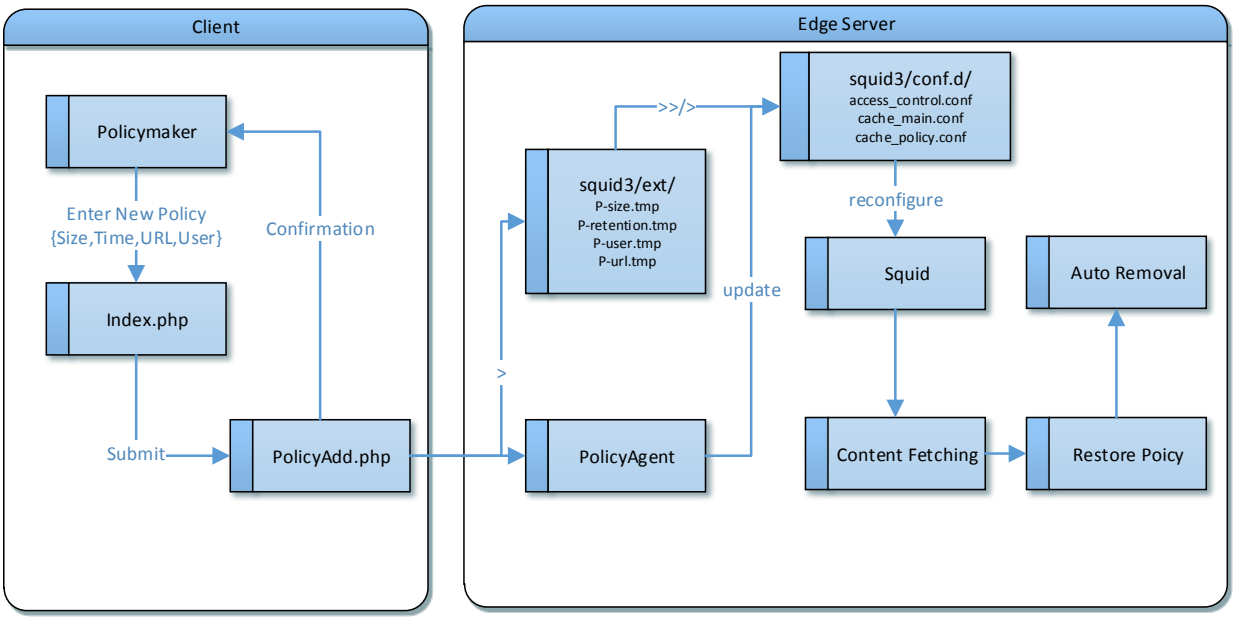


Figure 9 Flow chart of the solution realization.

5.4 Publisher Web Interface

We use simple PHP pages to interact with the user. The file contains a combination of HTML tags and PhP scripts to run commands on a Web server. We created a user-friendly Web page, allowing users to add new policy and submit the new policy to the server, while involving the server automatic policy update procedure which includes:

- Write new policy into TMP file;
- Apply new policy by creating new cache dir and lock other caches;
- Fetching intended content from network;
- After the download is completed, restore previous configurations and lock the dedicated cache;
- Apply access control policy;
- Start auto expiration check and removal.

The implementation involves two simple PHP files:

1) index.php: User Policy Input Main Page to allow a user type in the new policy;
2) AddPolicy.php: Policy Response Page to actually execute commands at local and remote servers and display output results at the browser to confirm the result to the user.

6 CONCLUSION

This deliverable provided a first insight into the RIFE system architecture at a component level, while also providing a first overview of the main interfaces at a high level. As a particularly important element to the overall RIFE proposition of localizing communication for cost reduction purposes, we elaborated on the first insights into the design for a proxy-based edge cache, allowing for off-peak placement of content and the integration with future context-aware cache selection solutions.

In our future work, reflected in updates to D2.2 in future deliverables, we will focus on detailing the various interfaces towards a full specification. This will be aligned with our ongoing development efforts, which provide the necessary insights into the various protocols, the byte formats and other detailed information for such specification.

7 REFERENCES

[Alz2015] B. Alzahrani, M. J. Reed, J. Riihijärvi and V. G. Vassilakis, "Scalability of information centric networking using mediated topology management," Journal of Network and Computer Applications, vol. 50, 126–133, 2015.

[ERI2015] Ericsson, "Mobility Report," June 2015. Available online: http://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf

[Fot2010] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing Information Networking Further: From PSIRP to PURSUIT," *BROADNETS, volume 66 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 1-13, Springer,* 2010.

[IET1981] IETF, "RFC 791: Internet Protocol, Protocol Specification," 1981. Available online: https://tools.ietf.org/html/rfc791

[IET1998] IETF, "RFC 2460: Internet Protocol, Specification," 1998. Available online: https://tools.ietf.org/html/rfc2460

[ITU2001] ITU-R, "BS.1387: Method for objective measurements of perceived audio quality," 2001. Available online: http://www.itu.int/rec/R-REC-BS.1387/en

[ITU2008] ITU-R, "P.910 : Subjective video quality assessment methods for multimedia applications," 2008. Available online: http://www.itu.int/rec/T-REC-P.910/en

[Jok2009] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar and P. Nikander, "LIPSIN: line speed publish/subscribe internetworking," in Proc. of ACM SIGCOMM, 2009.

[Kar2012] T. Kärkkäinen, M. Pitkänen, P. Houghton, and J. Ott, "SCAMPI Application Platform," In Proc. of ACM CHANTS, 2012.

[Poi2015] D2.1 Scenarios Requirements Specifications.

[RFC4838] V. Cerf et al, "Delay Tolerant Networking Architecture", RFC4838, 2007.

[RFC5050] K. Scott and S. Burleigh, "Bundle Protocol Specification", RFC5050, 2007.

[Tap2012] J. Tapolcai, A. Gulyas, Z. Heszberger, J. Biro, P. Babarczi and D. Trossen, "Stateless multi-stage dissemination of information: Source routing revisited," In Proc. of IEEE Globecom, 2012.

[Tro2010] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an information-centric internetworking architecture," *SIGCOMM Computer Communication Review, vol.* 40, no. 2, pp. 26-33, 2010.

[Tro2011] D. Trossen (ed.) et al, "Architecture Definition, Components Descriptions and Requirements," PURSUIT Deliverable D2.3, 2011. Available online as: http://fp7pursuit.ipower.com/PursuitWeb/wp-content/uploads/2011/12/INFSO-ICT-257217_PURSUIT_D2.3_Architecture_Definition_Components_Descriptions_and_Requirements.pdf

[Tro2012] D. Trossen and G. Parisis, "Designing and Realizing an Information-Centric Internet", In IEEE Communications Magazine, Volume 50, Issue 7, pp. 60-67, 2012.

[Tro2015] D. Trossen, M. Reed, J. Riihijärvi, M. Georgiades, G. Xylomenos, N. Fotiou, "IP Over ICN – The Better IP?", Proceedings of EuCNC (European Conference on Networks

and Communications), Paris, France, June 29/July 2, 2015.

[Tsi2013] C. Tsilopoulos and G. Xylomenos, "Scaling Bloom filter-based multicast via filter switching," In Proc. of IEEE ISCC, 2013.