Action full title:

Universal, mobile-centric and opportunistic communications architecture

Action acronym:

UMOBILE

Deliverable:

D4.2 “Flowlet Congestion Control – Final Report”

Project Information:

Project Full Title	Universal, mobile-centric and opportunistic communications architecture
Project Acronym	UMOBILE
Grant agreement number	645124
Call identifier	H2020-ICT-2014-1
Topic	ICT-05-2014 Smart Networks and novel Internet Architectures
Programme	EU Framework Programme for Research and Innovation HORIZON 2020
Project Coordinator	Prof. Vassilis Tsaoussidis, Athena Research Center

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Deliverable Information:

Deliverable Number-Title	D4.2 Flowlet Congestion Control – Initial Report
WP Number	WP4
WP Leader	SENCEPTION
Task Leader (s)	UCAM
Authors	**ATHENA:** Sotiris Diamantopoulos, Christos-Alexandros Sarros, Vassilis Tsaoussidis **UCL:** Ioannis Psaras, Sergi Rene **UCAM:** Adisorn Lertsinsrubtavee, Carlos Molina-Jimenez
Contact	i.psaras@ucl.ac.uk, s.rene@ucl.ac.uk
Due date	M30: 31/07/2017
Actual date of submission	31/07/2017

Dissemination Level:

PU	Public	x
CO	Confidential, only for members of the consortium (including the Commission Services)	
CI	Classified, as referred to in Commission Decision 2001/844/EC	

Document History:

Version	**Date**	**Description**
Version 0.1	27/07/16	First draft to the consortium of the flowlet congestion control
Version 1.0	31/07/16	Final submitted version of the flowlet congestion control

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

List of Definitions

Term	Meaning
DTN	Delay Tolerant Network (DTN) is an emerging technology that supports interoperability of other networks by accommodating long disruptions and delays between and within those networks. DTN operates in a store-and-forward fashion where intermediate nodes can temporary keep the messages and opportunistically forward them to the next hop. This inherently deals with temporary disruptions and allows connecting nodes that would be disconnected in space at any point in time by exploiting time-space paths.
ICN	Information-Centric Network (ICN) has emerged as a promising solution for the future Internet's architecture that aims to provide better support for efficient information delivery. ICN approach uniquely identifies information by name at the network layer, deploys in-network caching architecture (store information at the network node) and supports multicast mechanisms. These key mechanisms facilitate the efficient and timely information (contents and services) delivery to the end-users.
Content	Content refers to a piece of digital information that is disseminated and consumed by end-user equipment.
Node	A wireless or wired capable device.
User	An entity (individual or collective) that is both a consumer and a relay of user services.
User Service	Context-aware services are considered as a set of mechanisms that assist incorporating information about the current surrounding of mobile users in order to provide more relevant of services.
User Interest	A parameter capable of providing a measure (cost) of the “attention” of a user towards a specific (piece of) information in a specific time instant. Particularly, users can cooperate and share their personal and individual interests that enable the social interactions and data sharing across multiple users.
User Requirement	User requirement corresponds to the specifications that users expect from the application.
Upstream	Upstream refers the path toward the producer node.
Downstream	Downstream refers the path toward the consumer node.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Gateway	Gateway typically means an equipment installed at the edge of a network. It connects the local network to larger network or Internet. In addition, gateway also has a capability to store services and contents in its cache to subsequently provide local access communication.
Customer Premises	Customer Premises relate to residential households and enterprise market and are, as of today, controlled by the end-user.
User-centric	User-centric refers to a new paradigm leveraging user information at large to deliver novel content or services by users towards other users.
UMOBILE System	UMOBILE System refers to an open system that provides communication access to users through wired or wireless connectivity. This system exploits the benefit of local communication to minimize upstream and downstream traffic. The services or contents can be exchanged and stored in several devices such as gateways; user equipments; customer premises equipments such as WiFi Access Points in order to efficiently delivery the desired contents or services to end-users.
UMOBILE Architecture	A mobile-centric service-oriented architecture that efficiently delivers contents and services to the end-users. The UMOBILE architecture integrates the principles of Delay Tolerant Networks (DTN) and Information-Centric Networks (ICN).
User-equipment	User-equipment (UE) corresponds to a generic user terminal (for example smart phone or notebook). In terms of UE and for operating systems we consider mainly smartphones equipped with Android; notebooks with UNIX, Windows, Mac OS.
Social Trust	Trust which builds upon associations of nodes is based on the notion of shared interests; individual or collective expression of interests; affinities between end-users.
Application	Computer software design to perform a single or several specific tasks, e.g. a calendar and map services. In UMOBILE, context-aware applications are considered.
Incentive	A factor (e.g., economic or sociological) that motivates a particular action or a preference for a specific choice.
Service	Service refers to a computational operation or application running on the network which can fulfil an end-user's request. The services can be hosted and computed in some specific nodes such servers or gateways. Specifically, service is normally provided for remuneration, at a distance, by electronic means and at the individual request of a recipient of services. For the purposes of this definition; *"at a distance"* means that the service is provided without the parties being simultaneously present; *"by electronic means"* means that the service is sent initially and received at its destination by means of electronic equipment for the processing (including digital compression) and storage of data, and entirely transmitted, conveyed and received by wire, by radio, by optical means or by other electromagnetic means; *"at the individual request of a recipient of services"* means that the service is provided through the transmission of data on individual request. Refer to D2.2 for further details.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Trust Association	An unidirectional social trust association between two different nodes.
UMOBILE gateway	Role (software functionality) which reflects an operational behavior making a UMOBILE device capable of acting as a mediator between UMOBILE systems and non-UMOBILE systems – the outside world.
UAV	Unmanned Aerial Vehicle, which is an aircraft with no pilot on board.
BSS	Basic Service Set is a set consisting of all the devices associated with a consumer or enterprise IEEE 802.11 wireless local area network (WLAN). The service set can be local, independent, extended or mesh. Service sets have an associated identifier, the Service Set Identifier (SSID), which consists of 32 octets that frequently contains a human readable identifier of the network.
DC	Data Cache is responsible for holding information concerning the content carried by the current node.
SWCDG	Social Weight and Carried Data Gatherer is responsible for obtaining the list of interests and social weights towards the encountered node.
DM	Decision Maker is responsible for deciding whether replication should occur based on the level of social interaction towards specific interests, based on the SCORP algorithm.
SC	Service Controller manages the mapping of publishers of services and subscribers of services in the service migration module.
SP	Service Publisher refers to the original content producer in the the service migration module.
SEG	Service Execution Gateway is the point of attachment for clients in the the service migration module.
FN	Forwarding Node is responsible for routing requests for services towards the available copies in the service migration module.
NDN	Named Data Networking
PIT	Pending Interest Table
FIB	Forwarding Information Base

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Executive Summary

This deliverable (D4.2) is part of the five produced deliverables by WP4 that deal with QoS. In WP4 we address the challenges of QoS with congestion control mechanisms that enhance QoS by means of avoiding congestion problems that might result in packet loss, latency and low throughput.

The UMOBILE project devises different mechanisms to provide QoS to users. In the task 4.1, and described in D4.4 [1], we developed a set of mechanisms including service migration, congestion control and delay-tolerant mechanisms.

In this document we define the specifications of the new flowlet congestion control called In-Network Resource Pooling Protocol (INRPP), outcome of the UMOBILE project. D4.1 [2] discusses the first version of a congestion control protocol that addresses congestion control in TCP/IP networks, and in D4.2 we define the specifications of the flowlet congestion control INRPP over NDN networks, that will be integrated in UMOBILE networks, as specified in D4.4, to provide QoS.

In D4.2 we also a discussion about how we achieve the design issues identified in D4.1. A broad performance evaluation through simulations to assess the INRPP mechanisms and to quantify the improvement of INRPP over other congestion control protocols will be included in D5.2 (second validation methodology and evaluation report).

D4.2 is based on deliverables D2.1[3], D2.3[4], D3.2[5] and D3.4[6]. Deliverables D2.1 and D2.3 describe the requirements of the end-user and the system, respectively. On the other hand, D3.2 and D3.4 cover the UMOBILE architecture.

- The methodology used in this deliverable is as follows:
- We identify congestion control limitations that can be improved using in- network resources.
- We analyse existing work for congestion control in NDN networks.
- We detail a set of mechanisms required for the INRPP to implement the new congestion control over NDN networks

The document is organized as follows: Section 1 introduces core concepts of flowlets congestion control over UMOBILE framework. Section 2 presents the related work of congestion control in ICN networks. Section 3, provides details of INRPP flowlet congestion control design. In Section 4, we argue how we deal the INRPP implementation issues identified in D4.1 for the UMOBILE project. Finally, in the Conclusion section we summarize our findings and discuss open questions that we are currently addressing.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

1. Introduction

In the UMOBILE project (specified in the deliverable D3.4) we devise different UMOBILE devices (see Figure 1) that will perform in the UMOBILE domain. In the wired part of the UMOBILE domain, UMOBILE-enabled routers are specific devices that connect the Service Manager (part of the Service migration) and the UMOBILE gateway (detailed in D5.3[7]) with the UMOBILE hotspots. UMOBILE-enabled routers provide the Service migration service and the QoS mechanisms devised in this project. The difference between UMOBILE-enable routers and Internet routers is that UMOBILE-routers are supporting the UMOBILE protocols (based on NDN) and are compatible with the UMOBILE QoS mechanisms.

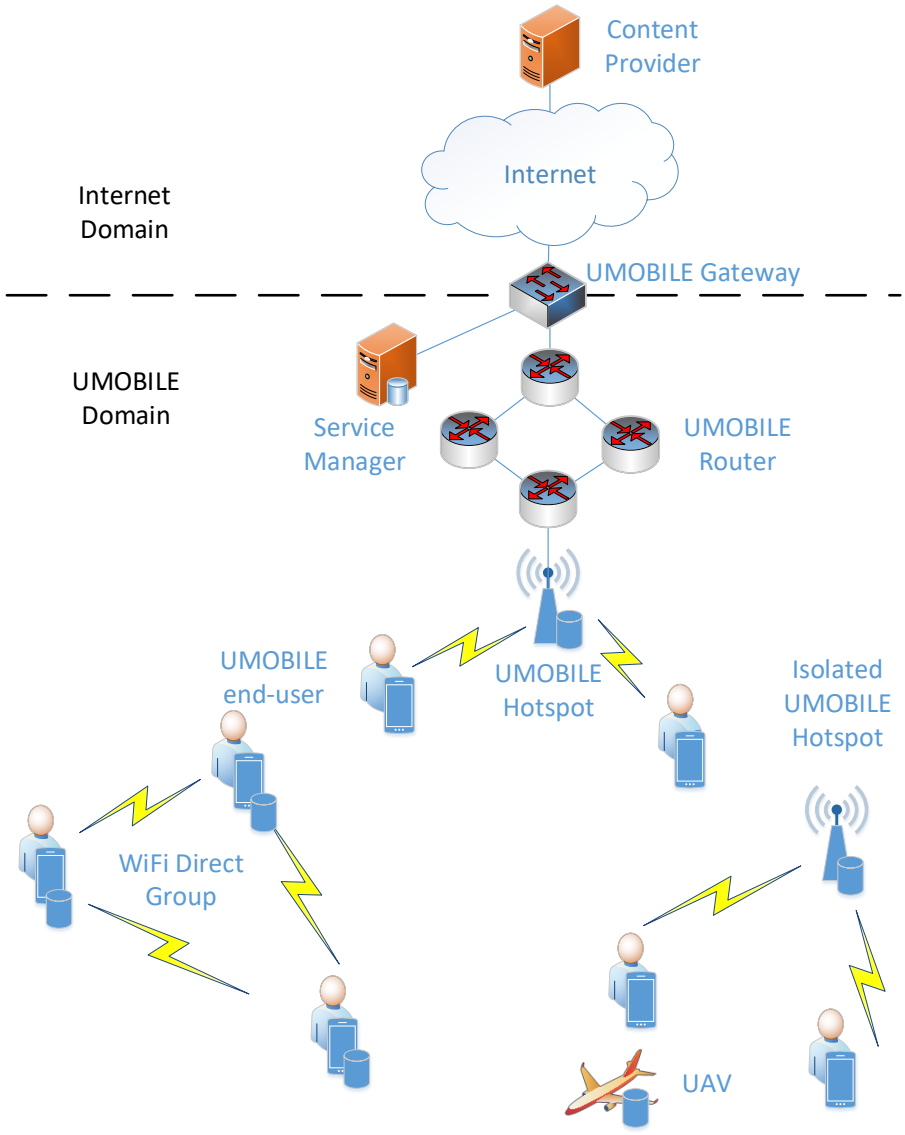

FIGURE 1 - UMOBILE ARCHITECTURE

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Among the QoS mechanisms mentioned there is the new flowlet congestion control mechanism that we will develop in this document. We focus in the flowlet congestion control based on INRPP for NDN networks. In this document, we use the concept flowlet instead of flow, since in ICN the destination is known but the source can be multiple nodes (caches) in the network. Therefore, content object can be retrieved from multiple nodes using several flowlets.

Congestion control has been considered as the topic of endless research ever since the standardisation of the Internet's main transport protocol, the Transmission Control Protocol (TCP). TCP acts as the main transmission chain between any two endpoints in the Internet, transmitting data with strict end-to-end feedback controls, in order to avoid congestive collapse [8]. Although TCP's main principles have kept the Internet running without major congestion events for decades, numerous performance issues have been repeatedly identified (e.g., [9,10]). Solutions to these problems [11, 12, 13, 14] have continuously and stubbornly been rejected mainly due to fear of instability.

There are two main uncertainty factors that fuel fear of instability and with which any reliable congestion control protocol has to deal with: i) the input load factor: the network does not know how much data the senders will put in the network next, and ii) the demand factor: there might be excessive demand for bandwidth over some particular area/link. TCP, for instance, defends against the input load factor through the Additive Increase/Multiplicative Decrease transmission model [15,16], while it deals with the demand factor by adopting the "one-out, one-in" packet transmission principle (only when a packet gets out of the network is a new one allowed in). Those two mechanisms are closely linked and interrelated and lead to TCP's defensive behaviour by effectively (proactively) suppressing demand. In essence, end-points have to speculate on the available resources along the end-to-end path and move traffic as fast as the path's slowest link.

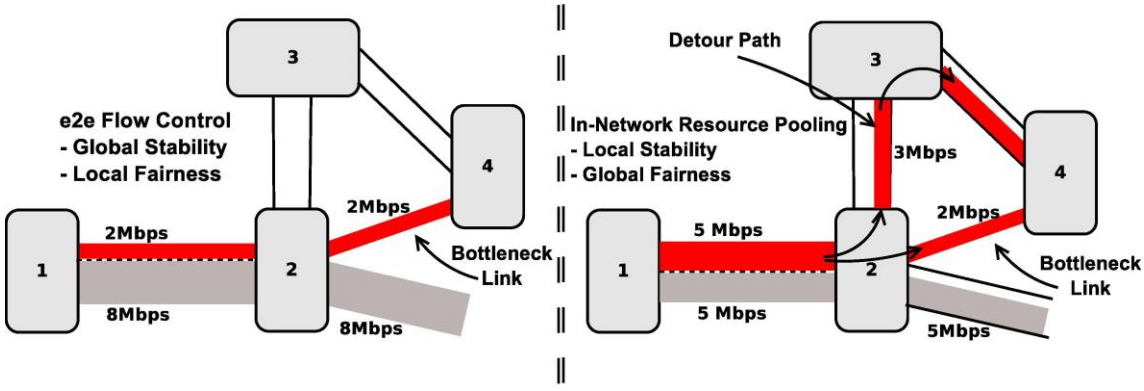

FIGURE 2 - LEFT: E2E FLOW CONTROL: BANDWIDTH IS SPLIT ACCORDING TO THE SLOWEST LINK ON THE PATH. RIGHT: INRPP: BANDWIDTH IS SPLIT EQUALLY UP TO THE BOTTLENECK LINK (GLOBAL FAIRNESS). DETOUR APPLIES TO GUARANTEE LOCAL STABILITY.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Given the single-path nature of TCP, moving traffic according to the path's slowest link guarantees global stability (i.e., stability along the *e2e* path through *e2e* rate-adaptation). Fairness on the other hand, is guaranteed locally (i.e., based on the capacity of the bottleneck link). We argue against this relationship and in the spirit of INRPP propose that: i) stability should be local, and ii) fairness should be global. Local stability demands that the node located next to the bottleneck link could take appropriate action when conditions deteriorate. Global fairness on the other hand requires that all resources up until the bottleneck link are shared equally among participating flows. Consider two flows in the topology of Figure 2. According to the *e2e* flow control of TCP (left part), the flow that traverses the bottleneck link (2-4) would achieve 2Mbps throughput (global stability), while the second flow would dominate the shared link (1-2) and achieve 8Mbps throughput. According to Jain's Fairness Index [17], given by $F = \frac{\Sigma(T)^2}{n\,\Sigma(T^2)}$, where T is each flow's throughput and n is the total number of flows, the system fairness in this case is 0.73. In case more than one flows traverse the bottleneck link (2-4), they would share equally the available bandwidth (local fairness). In contrast, according to the global fairness, the shared link (1-2) is split equally among the two flows. Node 2 has two options in this case: i) find alternative routes to reach node 4 (local stability), or ii) notify node 1 to reduce its sending. In the topology of Fig. 2, node 3 can accommodate the extra 3Mbps. In this case, Jain's index indicates perfect system fairness equal to 1.

Within the UMOBILE project, we aim to design and evaluate the In-Network Resource Pooling Protocol (INRPP) [18], which pools bandwidth and in-network cache resources in a novel congestion control framework to reach global fairness and local stability. Taking profit of the hop-by-hop design and the caching capabilities inherent in the NDN networks, or adding caches (i.e., temporary storage) and breaking the end-to-end principle, we argue that the demand factor can be tamed. Given this functionality of in-network storage, INRPP comprises three different states: i) push: content is pushed as far in the path as possible in an open-loop, processor sharing manner [19], based on the path's hop-by-hop bandwidth resources to take advantage of under-utilised links; ii) store and detour: when pushed data reaches the bottleneck link, the excess data is cached and simultaneously forwarded through detour paths towards the destination; iii) backpressure: if detour paths do not exist or have insufficient bandwidth, the system switches to the backpressure state [20, 21] to avoid overflowing of the cache. During the backpressure state, the nodes enter a closed-loop mode, where an upstream node sends one data packet per one received ACK to the backpressuring downstream node.

INRPP inherently prerequisites availability of alternative sub-paths (to allow for detours) and in-network content caches. While adding cache capacity in routers requires extra investment, the cost of the memory itself has fallen to affordable levels and we therefore, assume this not to be a prohibitive factor. That said, in order to prove the feasibility of

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

having one or more detour paths, in the Internet domain, to divert excessive traffic at the router level, we analysed a set of real topologies (from Rocketfuel [22]) for nine ISPs (see Table 1). Indeed, we find that six out of the nine real network topologies analysed can provide at least one 1-hop detour path on more than 50% of links, reaching up to 92.3% for Level-3 topology (second column). Columns 3-7 show how the percentage of 1-hop detour paths (shown in column 2) is split between 1-, 2-, 3-, 4- and 5+-hop detour paths. For instance, out of Telstra's 68.75% of paths that have a 1-hop detour, 7.9% have four 1-hop detour sub-paths along the main path. Overall, we see that, in most cases, when a detour path exists, there is more than one detour sub-path along the same edge-to-edge path. The extreme case of the Level3 network shows that 47.06% of its edge-to-edge paths with at least one detour sub-path have five or more 1-hop detour sub-paths. The final column in Table 1 is the maximum number of 1-hop detour sub-paths that the topology has for at least one of its links. Overall, we observe that networks are rather well-connected and would realistically allow for detouring of excessive traffic mid-path in the network.

Network	1-hop detour	Number of detour paths (% of col. 2)					Max 1-hop
		1	2	3	4	5+	
Telstra	68.75 %	27.45	30.09	11.5	7.9	23.06	38
Sprintlink	57.3 %	44.9	20.7	14.9	6.8	57.6	27
Ebone	51.8 %	55.5	28.78	10.6	3.53	1.5	10
Verio	71.75 %	23.56	18.01	13.45	12.56	32.52	25
Tiscali	24.44 %	60.60	19.19	12.12	5.05	3.03	8
Level3	92.30 %	13.40	14.10	12.42	13.02	47.06	95
Exodus	50.33 %	50.67	17.93	16.59	8.96	5.82	6
VSNL	25 %	100	0	0	0	0	1
AT&T	34.84 %	56.07	17.68	11.88	4.7	9.67	24

TABLE 1 – DETOUR PATHS IN REAL TOPOLOGIES

In this version of the flowlet congestion control deliverable we present the complete design, including the framework overview, the different mechanisms used by the solution and an initial performance evaluation of the INRPP congestion framework over NDN networks.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

2. Congestion control in ICN networks

Since, the UMOBILE domain relies on the NDN transport protocol which is different from the traditional TCP/IP, the existing TCP congestion control algorithms cannot be applied directly in the UMOBILE network. For instance, NDN follows the connection less and multi-source communication where the desired Data packets can be retrieved from many sources. As such some congestion detection mechanisms like Retransmission Time Out (RTO) may not be feasible in NDN, since it requires a single source-path communication between source and destination nodes [23]. However, congestion control algorithms over NDN can take advantages of ICN features.

In NDN, routers can manage traffic load through managing the Interest forwarding rate on a hop-by-hop basis; when a router is overloaded by incoming data traffic from any specific neighbour, it simply slows down or stop sending Interest packets to that neighbour. This also means that NDN eliminates the dependency on end hosts to perform congestion control. Once congestion occurs, data retransmission is aided by caching since the retransmitted Interest will meet the Data right above the link the packet was lost, not the original sender. Thus, NDN avoids congestion collapse that can occur in today's Internet when a packet is lost at the last hop and bandwidth is mostly consumed by repeated retransmissions from the original source host. In addition, the NDN routers can benefit from the use of PIT entry while managing the traffic load through the size of PIT entry [24]. As each PIT entry records a pending Interest request that has been forwarded, waiting for the Data message to return. Each NDN router can directly control the rate of traffic by controlling the rate of forwarding Interest. These inherent features of NDN are very useful for hop-by-hop congestion control. This feature clearly benefits the In-Network Resource Principle that aims at breaking the e2e congestion control to provide a hop-by-hop congestion control. Using INRPP we can use inherent caching features as a temporary custodian to deal with congestion locally, detouring traffic when other paths are possible and doing backpressure to the sender node when no more traffic can be allocated.

Relevant to INRPP is the discussion of Flow Classification in ICN presented in an informal Internet draft [25] submitted recently by Moiseenko and Oran from Cisco Systems. They suggest two mechanisms (Equivalent class component count and Equivalent class name component type) for ICN flow identification that can be potentially used, among other functionalities, to support congestion control. The key idea is to identify flows by means of the name prefixes of the Interest and Data packets exchanged between consumers (receivers) and producers (senders): equivalent classes of flow are associated to the names in the corresponding Interest and Data packets. In [26] the authors discuss RRCP--- a congestion control protocol for designed specifically for ICN networks and based on the XCP (eXplicit Control Protocol) presented in [27]. Though the authors show only simulation results, they argue that their solution is promising as it takes advantage of the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

main features of ICN transport like consumer-driven data transfer and in-network caching. A comprehensive survey on congestion control in ICN is presented in [28].

Recent efforts in the ICN transport-layer area have mainly focused on adjusting the main congestion control mechanisms of TCP and AIMD to fit to an ICN environment (e.g., [29-35]). Closer to our INRPP work are [32], [33] and [35]. Although the protocol in [35] uses detours to find less utilised links (similar to the concept of INRP), it then deploys AIMD over single paths to regulate the sending rates, hence, adopts the drawbacks of TCP. Furthermore, detouring in [35] takes place at the Interest phase (instead of the data phase), hence, its accuracy is bound to be outdated by approximately. The work in [33] on the other hand, is based on hop-by-hop rates to shape the rate of interests and therefore, data as well (similarly to [36]). Note that none of the above ICN-oriented transports has been evaluated together with caches, something that completely rules out the benefits of in-network storage and limits the full potential of the ICN paradigm.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

3. INRPP over NDN networks

Given the dynamic nature of the considered use-cases, UMOBILE does not rely on the existence of end-to-end flows between communicating parties. Thus, the concept of flowlets is rather interesting, to provide applications with different quality of service mechanisms to allow satisfactory levels for the content being exchanged. This functional block oversees how flowlets are started between the involved entities as to allow better, QoS-based data exchange and provide special support to in-network caching. It may provide QoS on-demand, by understanding the needs for QoS and making sure the content reaches its recipients respecting the desired QoS levels as much as possible.

One can use congestion control mechanisms to prevent congestion problems that might result in packet loss, latency and low throughput. Congestion control mechanisms address QoS by means of manipulating information at network level such as monitoring of queues with subsequent manipulation of network packet within the routers. Examples of these techniques are traffic management by means of end-to-end or hop-by-hop congestion control.

Taking profit of the hop-by-hop design and the caching capabilities inherent in the NDN networks, or adding caches (i.e., temporary storage) and breaking the end-to-end principle in TCP/IP networks, we argue that the demand factor can be tamed, providing global fairness and local stability. Given this functionality of in-network storage, INRPP comprises three different modes of operation, depicted in Figure 3:

- Push: content is pushed along the path between Data producer and Data consumer in an *open loop* mode. The data sending rate is adjusted based on the neutral processor sharing rate between producer and consumer. The bulk data transfer is also applicable in this mode to take advantage of under-utilised links along the path.
- Store and Detour: when pushed data reaches the bottleneck link, the excess data is cached and simultaneously forwarded through detour paths towards the destination;
- Backpressure: if detour paths do not exist or have insufficient bandwidth, the system switches to the backpressure mode of operation to avoid overflowing of the cache. In the backpressure mode, the nodes enter a *closed loop* state, where an upstream node sends one data packet per one received ACK to the backpressuring downstream node.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

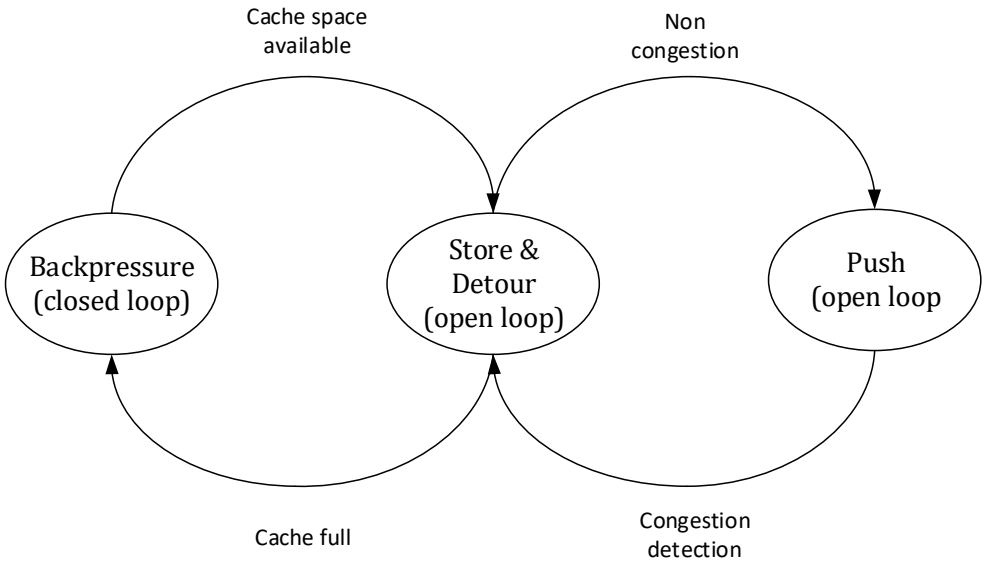

FIGURE 3 – INRPP STATE DIAGRAM

Note that congestion control mechanisms are based to the optimisation of network resource usage, consequently, their effectiveness depends on and is bound by the amount of resources available. In other words, congestion control mechanisms can do nothing when the network resources are already exhausted. In that case, application-level QoS mechanisms are required. In D4.4 [1], we present the service migration mechanism to support the QoS in application level.

In summary, INRPP congestion control features the following characteristics to provide QoS network-level mechanisms:

- Hop-by-hop congestion control using in-network caches as a temporary custodian to alleviate temporary congestion without slowing down the source.
- Use of in-network multipath, being able to use unused bandwidth in one-hop detour paths (i.e., paths that can be used to reach the next node only with a difference of a one hop more).
- It provides network stability using backpressure mechanisms.
- Improves flows completion time and better use of resources.
- Using differentiated level services in the caches, we can add priorities to deliver packets from the temporary custodian cache to the link at different rates.

In the following we specify the cache storage system in INRPP, how we detect congestion in local links, the detouring mechanism and an illustrative example of how the backpressure mechanism works.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

3.1 Cache storage system

INRPP uses in-network storage for senders to push as much data as possible into the network and deal with congestion within the network as opposed to the edges. As in any cache implementation, INRPP caches maintain an indexed flow table and insert in-coming packets in the order of arrival. This way we: i) efficiently retrieve data packets belonging to particular flowlets while performing pacing, ii) forward packets ordered by sequence (of arrival) to avoid reordering issues at the receiver, but most importantly, iii) avoid head of line blocking at the cache.

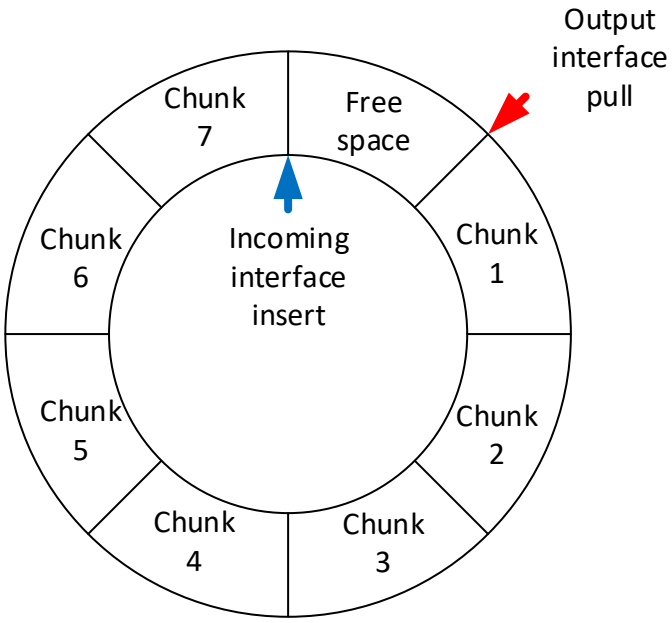

FIGURE 4 – CIRCULAR CACHE DESIGN

INRPP caches are indexed by: i) the primary output interface, that is, not the detour interface, if any, and ii) the content object id. As object id, we use the name prefix the object discarding any naming element aimed at chunks sequencing.

INRPP caches can be seen as a circular buffer, as stated in Figure 4, where the content chunks are inserted when the packets arrive to the router (since we index the cache per output interface, is like having a separate logic cache for each one). Outgoing interfaces pull packets from the cache at the rate determined by the link rate, unless the router received a slow-down notification from this interface, meaning that it needs to pace data packets to the bottleneck available rate. In that case, the router fetches a data packet from the cache corresponding to that flowlet only if the determined rate allows it. Next, we discuss how INRPP nodes discover the available residual bandwidth on detour paths.

3.2 Detecting congestion

In order to detect congestion, we measure the link delay locally, instead of trying to infer congestion at the edge of the network by detecting packet loss or monitoring RTT. To

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

detect congestion on router, we apply a well-studied AQM mechanism to detect congestion in our cache system design, based on delays [8]. The basic idea is to measure queueing delay ("sojourn time") in the cache to detect that the outgoing interface is not sending packets fast enough to absorb the incoming rate. If the minimum sojourn time over a time period (i.e., 100ms) exceeds a certain threshold (target link delay -default: 5ms-), we consider the link is congested. At this point, the output interface switch to *Store&Detoure (S&D)* and starts sending some of the packets towards the output interface using a detour interface to the same destination. In the following section, we describe how the detouring mechanism balance some traffic when congestion to uncongested detour paths. Alternatively, when the minimum sojourn time over the specified time period exceeds the defined threshold, the congestion interface switches back to the *Push* mode and stops detouring.

Additionally, we can measure congestion monitoring the PIT size. A PIT that is growing too large and threatens to overflow the router's memory, can also be signaled downstream to reduce the number of interests and therefore avoid congestion when the PIT size reach a determined threshold. In that case, the consumer reduces its sending rate immediately, without waiting for the backpressure mechanism being propagated.

To notify congestion to one-hop routers we use a special Interest (similar to the Interest used to create faces internally in the NDN Forwarding Daemon -NFD-), that is sent towards the directly connected nodes using the congested link. This way, the downstream node can detect the previous node has detected congestion and is in *S&D mode,* and can start detouring some Interests to the detour path interface.

3.3 Detouring mechanism

In INRPP, a node detours its traffic in order to eliminate the excess traffic stored in its cache through alternative paths. In order to avoid severe detour delays and packet reordering, nodes refrain from using already congested detour paths as sending traffic to a congested interface would result in caching of the detour traffic along the detour path.

INRPP makes use of a simple link-state protocol in order to help nodes identify the neighbours with which they are connected through a 1-hop detour path. In the example of Figure 5, node r determines that its next-hop neighbours d and n are directly connected by examining their link state advertisements. The link-state protocol would be slightly more complicated for multi-hop detour paths, which we do not cover in this document.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

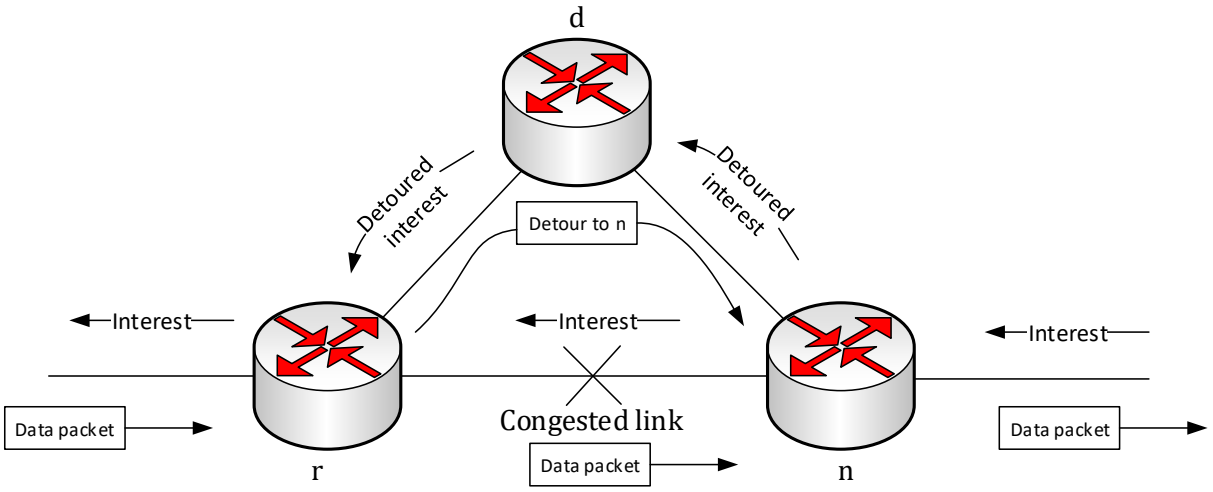

FIGURE 5 – INRPP ROUTER AND DETOUR EXAMPLE

Consider the example topology in Figure 5 where a router r has a one-hop detour path through d to reach n. When r's interface facing n is congested, the interface switches to *S&D* mode. At this point, meanwhile r is caching excess data, it notifies n *to* start demultiplexing some of the interests between the primary outgoing interface and the detour interface(s) as shown in Figure 5. The router sends interests, corresponding to data packets downstream, only if the path has residual, i.e., unused, bandwidth to forward the traffic. We determine that a path has residual bandwidth available if the detour path node has not notified congestion as well. If this is the case, we immediately stop sending interests through this interface.

To determine the amount of traffic that we want to divert using detour paths, and therefore the amount of interests to send towards the detour path interfaces, we believe a good solution is similar to the solution used in [37] to multiplex some traffic in case of congestion. For each congestion signal (sent each 100ms by default if the congestion persists), it reduces the forwarding percentage of the incoming interface, while at the same time increasing the forwarding percentage of all the available detour paths to an equal amount, determined by the following equations:

$$reduction = fwPerc(P) * CHANGE\_PERC \quad (1)$$

$$fwPerc(P) - = reduction \quad (2)$$

$$fwPerc(D) + = reduction \,/\, num\_detours - 1 \quad (3)$$

where *reduction* is the amount of traffic detoured, CHANGE_PERC is a fixed parameter between 1% and 3%, *fwPerc(P)* the traffic sent through the main path, *fwPerc(D)* the traffic sent using the detour paths and *num_detours* the number of detour paths available. The *CHANGE_PERC* is selected between 1% and 3% in [37] because it works well for several different bandwidths. In our case, we will report some verified evaluations in D5.2 INRPP to check whether INRPP is performing well using these values.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

In case there is no exist any detour path, no traffic is detoured and the backpressure mechanism, detailed in the next section, is triggered when the cache reach its maximum capacity (i.e., starts overwriting data chunks that are not send).

3.4 Backpressure

In the context of INRPP, it is important to distinguish the router's *interface mode* (discussed above) from the *flowlets' states*. Generally speaking, INRPP flowlets operate in either *open loop* or *closed loop* state. However, the fact that some flowlets can be affected by the bottleneck of the link while other flowlets can go towards an unaffected part of the network, results in the situation where different flowlets are in different states in the same node. In Figure 6 we give an illustrative example how *closed loop* state is propagated when different nodes activate the backpressure mode.

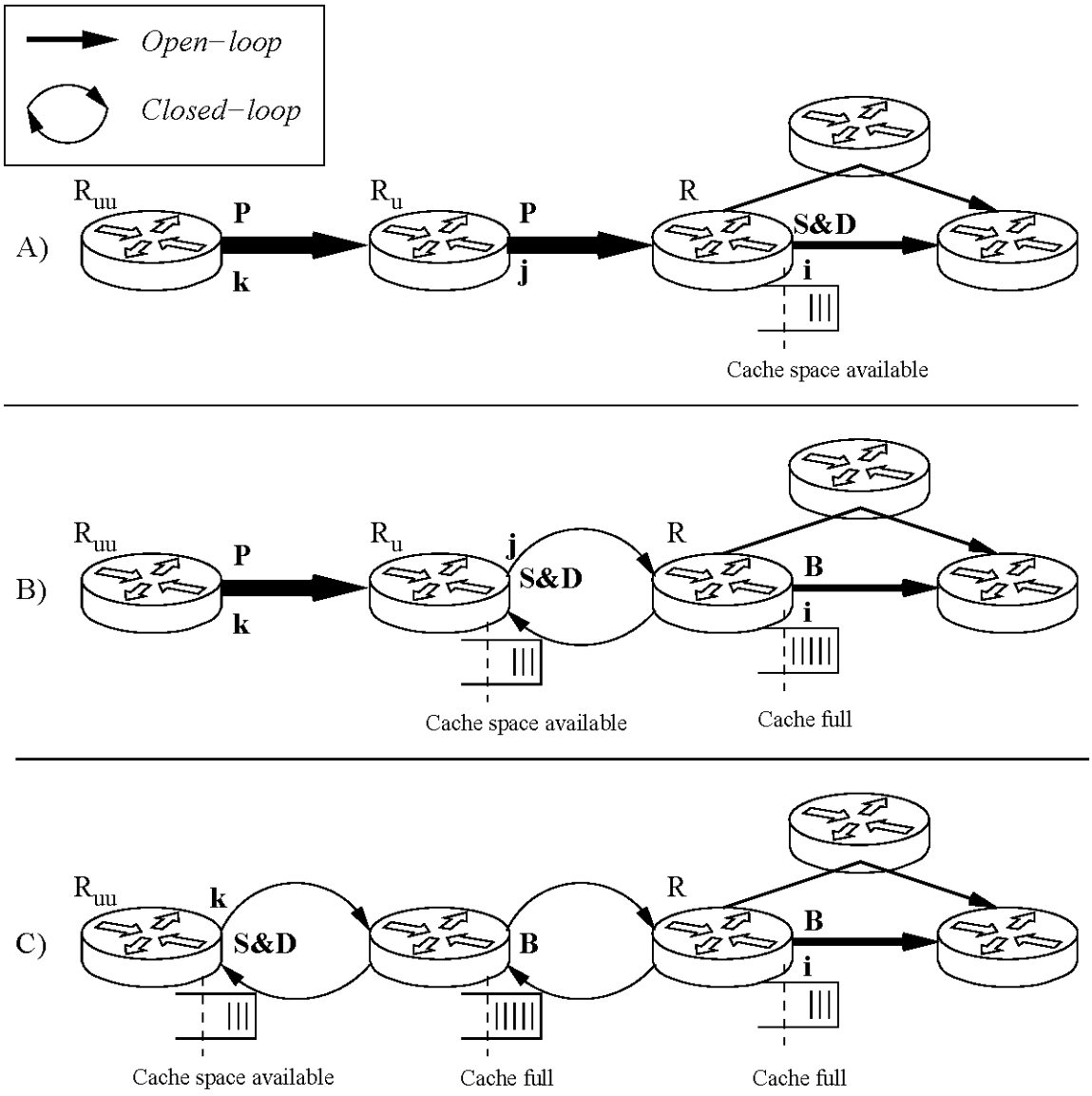

FIGURE 6 – HOP-BY-HOP STATE PROPAGATION

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

Similar to TCP's slow-start, the new flowlets mechanism enters the system in an open loop state at the sender. Data consumers will request data sending interests at the application rate, and for bulk data transfers an initial processor sharing rate must be set.

As a first step to implement INRPP over NDN, we assume Interest packets and Data or Content chunks generated by consumers and producers, respectively, without breaking the one data packet by one interest rule of NDN. The format of the Request Packets (Interests) is the same than NDN. However, we assume each chunk size is fixed. Data senders (producers) should keep state for each flowlet (similar to TCP senders) and operate in *closed loop* (bottleneck rate) or *open loop* (processor sharing rate).

NDN natively follows a synchronous communication model where a Data consumer sends one Interest message to retrieve one Data message in turn. If Data consumer waits until it receives the prior request data chunk before sending the Interest messages for subsequent data chunks, is not efficient and the link bandwidth is not fully. As mentioned in the previous section, the *open loop* state allows the bulk data transfer between Data producer and consumer. To support *open loop*, we use a multi-Interest forwarding model by allowing the Data producer sends an aggregation (burst) of *N* Data chunks towards the destination node at once. Figure 7 shows the message flows between Data producer and destination node using multi-Interest forwarding model. A Data producer initially sends a push Interest message including the information about the total content size (C_s), data chunk size (D_s) and available local link bandwidth (Bs). When the destination node receives the Interest message, it calculates the number of requests required to retrieve the whole content $Tn_c = \lceil C_s \ / \ D_s \ \rceil$. The destination node can start requesting chunks within the second request by setting a burst size *N* equal to the minimum local processor sharing rate between the producer and the consumer set up using the producer parameters, i.e. the maximum available bandwidth in the local link between consumer and producer. After receiving the first few chunks of data, the receiver will continuously send interests at the application or the agreed processing rate while its working in *open loop* state, unless the backpressure state is propagated back to the producer, that will notify the consumer to slow down its rate to the bottleneck available rate, changing to *closed loop* state.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

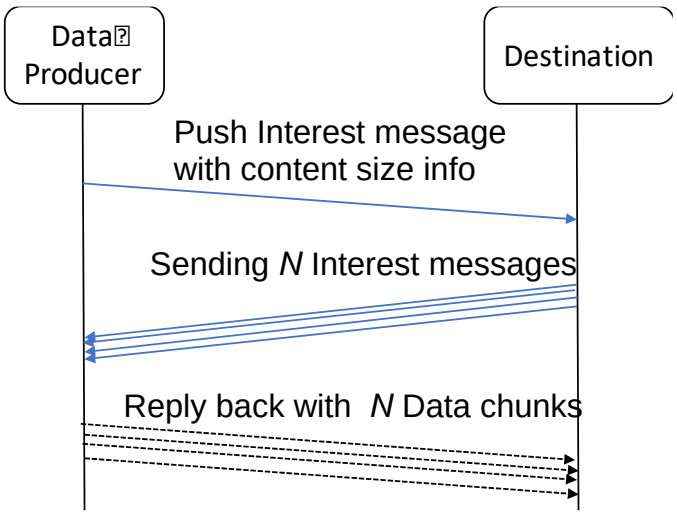

FIGURE 7 - DATA PRODUCERS PUSH MULTIPLE CHUNKS OF CONTENT TOWARD THE DESTINATION NODE WITH PRIOR NOTIFICATION OF NEW AVAILABLE CONTENT

Triggered by a slow-down notification originating from some backpressuring interface *i* along the path, flowlets that traverse *i* enter a *closed loop* state *between the current and the immediate upstream node*, that is, not along the entire path. The *closed loop* state extends further upstream when the cache of the immediate upstream node of the backpressuring router reaches its capacity.

To avoid propagating the backpressure mode to the producer, the node in backpressure rate starts marking interest packets at the rate determined by the congested link. For example, if the interest rate corresponds of 1Gbps of data, and the available rate ate bottleneck link (bottleneck + detour) is 500Mbps, the congested router starts marking half of the interests. This way, when the upstream router interface enters in *closed loop* mode, it slows down the rate to the rate allocated in the bottleneck to avoid overflowing the bottleneck, but it does not slow down the producer. Also, only the flowlets corresponding to the interests marked will slow down, to avoid slowing down other flowlets not affected by the bottleneck.

Once the upstream node cache is full, this node propagates again the *closed loop* state to the next hop, until it reaches the producer, that also slows down the rate when it enters *closed loop,* adapting its source rate to the bandwidth available in the network, but only for the flowlets affected by the bottleneck (i.e. the flowlets towards router *R* interface *i*).

Consider the example in Figure 6, where we demonstrate a sequence of state changes in three consecutive stages. Initially (see top part), the upstream nodes of router *R* (*Ru* and *Ruu*) send traffic to its interface *i* in an open-loop (i.e., Push) manner. Eventually, the occupancy of cache (at *R*) exceeds its capacity and enters *S&D* state (see middle part of Fig 5), i.e., *Ru* slows down the data packets rate towards *R*, sending at rate determined by *R*. Note that at this point, interface *j* is in *S&D* mode, while flowlets are in *closed loop* state downstream from *Ru*'s interface *j* and in *open loop* state upstream from *Ru*'s interface *j*. That is, the upstream node of *Ru* (*Ruu*) continues to send packets towards *R* in an open-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

loop manner because *Ru*'s *j* interface cache has not reached its cache capacity limit yet. To achieve this, *Ru* does not propagate the slow-down instruction upwards.

Once the cache occupancy of *Ru* also exceeds the its capacity, it enters Backpressure mode too, and sends slow-down messages upstream to *Ruu* to slow-down the traffic flowing through *j*. At this point, the flow(lets)' *closed loop* state extends to *Ruu* as shown in the bottom part of Figure 6.

The hop-by-hop flow states and their propagation is essential to avoid packet drops from INRPP caches. In case any packet is dropped because the cache capacity reach its limit and the output rate is not enough to send packets in the cache before are overwritten, interests are reissued upstream to be able to get the packet dropped without the necessity of reissuing the interest from the client

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

4. INRPP implementation identified issues in D4.1

In D4.1 [2] we briefly described some issues that we previously identified to implement INRPP over NDN for the UMOBILE domain. In the following we add a discussion about how we deal with these issues in the current NDN implementation of the INRPP flowlet congestion control:

Push services

Based on the scenario requirements, UMOBILE platform requires both pull and push-based communication models as mentioned in D3.3. In document D3.1 we previously proposed 3 different mechanisms to support push-based services over NDN. However, not all of them maintain 1-to-1 flow balance of Interest Packets and Data packets. Therefore, we cannot foresee the traffic over an interface by keeping track of the requested packets and we would need a secondary mechanism to detect congestion in a link. However, the monitoring method used in the INRPP implementation for TCP/IP cannot be used in case of wireless links, since frequent disconnections and packet loss in wireless environments, hinder the possibility of using the same monitoring mechanisms than used in Section 2.2.

In D3.4 [6] document we have specified that the mechanism selected in the UMOBILE project to push content, is still relying in the one-to-one Interest packet for each Data packet rule. Since INRPP is also satisfying the one-to-one rule, there is no issue regarding breaking that rule. Regarding how to foresee the traffic over an interface, the NDN INRPP version is not actively monitoring link traffic, but measuring delay ("sojourn time") in the caches used. Thus, we do not rely in complex monitoring tools, avoiding this issue.

Wireless links

In the INRPP solution for TCP/IP networks we assume the capacity of a link is known and constant. However, in highly dynamic and disrupted wireless environment, such as the UMOBILE domain, we cannot consider the capacity of a link as known and constant, since it is highly variable and depends on several factors, such as SNR, interference, contention, etc. Therefore, we will need a mechanism to foresee the capacity of a link in a determined time, and the state/condition of the link (connected, disconnected, packet loss, etc).

In this document, we assume the network is using wired links with determined and fixed bandwidth. Links are pulling packets from the cache at the link rate to avoid congestion. However, we can easily adapt and extend this mechanism to be able to perform in wireless links, getting some feedback from the link layer (wireless) and adapting the rate to the available rate at certain time in a hop-by-hop manner.

DTN opportunistic links

In the forwarding mechanisms, initially proposed in D3.1, we included an IBR-DTN Delay Tolerant Networking approach, where we enable delay tolerant communications. However, including DTN faces in NDN, hinders even more the implementation of the INRPP

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

mechanisms, since detouring and backpressure assumes undisruptive communications. So we need to further investigate whether detouring and backpressure mechanisms can still be used with DTN faces and how.

In the same way than the previous issue, DTN links can suffer from disconnections and available bandwidth can be unknown. However, the hop-by-hop behavior of the INRPP congestion control makes it suitable to work in DTN links and only send packets to disruptive links or detour paths when those are active and able to transmit packets to the next-hop node.

Out-of-order delivery

Out-of-order delivery is an important issue that we will need to be dealt with, in case of detouring path with different characteristics.

Out-of-order delivery is indeed a problem at the transport level, since a lot of reordering can imply large buffering times at the receiver, and therefore and added delay to the applications. However, since we do not rely for instance, in duplicate acknowledgments to detect congestion, out-of-order delivery is not a big issue for INRPP. Even though, we avoid detouring to more than one-hop detour paths, or detour paths with very different latencies to minimize buffering times at the receivers caused by out-of-order packets.

Pull multicast

Regarding the in-network caching feature of NDN, the content is automatically cached in the routers along the path between the content provider and the receiver (on path caching). Consequently, if there is a subsequent request for the same content, the router's cache will directly response to the request without adding more traffic to the network towards the original content provider. In this manner, the congestion will be significantly controlled as the bandwidth over the network is better utilised. To benefit from this feature, we aim to further investigate pull multicast mechanisms in situations where several users requests the same piece of data. A potential solution is to develop mechanisms to support request aggregation in the routers. The intuition is that upon receiving several requests for the same piece of data, the router abstracts them as a single request and sends it to the original content provider. When the corresponding response is received (either a single or several chunks) the router caches it in its memory. Then it multicasts it to the group of requesters.

NDN, by default, groups interest for the same content to a single interest, avoiding sending multiple interest for the same content more than once and returning a single copy of the content when there are a group of users interested in the same content. Therefore, NDN is a perfect architecture to provide multicast group communications, saving bandwidth and transmitting content more efficient. However, since there is no one-to-one flows, identified by the 5-tuple of src/dst IP address, src/dst port, and the transport protocol, NDN rises some questions about the concept of fairness and how to apply it. The existing concept of "per-flow fairness" does not directly apply to NDN. Here we define an NDN content flow by requests for a specific content object from one or

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

multiple consumers. In order to do so, INRPP caches index its content by content name prefix providing a sort of per content queuing, sharing the available bandwidth not per "one-to-one" flow, but per "multicast" group, meaning one-to-many content object transmission.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

5. Conclusions

In this document, we defined the NDN version of the flowlet congestion control called In-Network Resource Pooling Protocol (INRPP) that will be part of the UMOBILE architecture. In this document, we included a design of the protocol along with a discussion about previous work in ICN networks and how INRPP solves possible design issues.

UMOBILE flowlet congestion control takes advantage of the transparencies natively supported by ICN networks like NDN to ease the adaptation of the INRPP congestion control. As briefly mentioned in the Introduction, a key observation is that in data centric applications, data is not necessarily transmitted directly from the data producer to the end data consumer. It is very likely that the original data will eventually reach its end data consumer through a data pipeline composed of several intermediate data services that are responsible for performing certain operations on the data (for example, caching, formatting, aggregation, filtering, cleansing, etc.).

To prove the usefulness and the better performance of INRPP in contrast with other congestion control approaches in NDN, we are implementing INRPP over NDN using the ndnSIM simulator [38]. A broad performance evaluation through simulations to assess the INRPP mechanisms and to quantify the improvement of INRPP over other congestion control protocols will be included in D5.2 (second validation methodology and evaluation report).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

References

[1] Carlos Molina-Jimenez, Adisorn Lertsinsrubtavee, Sergi Rene, Ioannis Psaras, Sotiris Diamantopoulos, Christos-Alexandros Sarros, Vassilis Tsaoussidis, "D4.4 Set of QoS interfaces and algorithms," UMOBILE Project, July 2017.

[2] Sergi Rene, Ioannis Psaras, Sotiris Diamantopoulos, Ioannis Komnios, Vassilis Tsaoussidis, Adisorn Lertsinsrubtavee, Arjuna Sathiaseelan, Carlos Molina-Jimenez, "D4.1 Flowlet Congestion Control – Initial Report," UMOBILE Project, July 2016.

[3] Paulo Mendes, Sotiris Diamantopoulos, Nikos Bezirgiannidis, Ioannis Komnios, Ioannis Psaras, Sergi Rene, Adisorn Lertsinsrubtavee, Arjuna Sathiaseelan, Susana Perez Sanchez, Francisco Almeida, Francesco Amorosa, Giammichele Russi, Angela d'Angelo, Alberto, "D2.1 - End-User Requirements Report," UMOBILE project, Grant 645124, June 2015.

[4] Paulo Mendes, Sotiris Diamantopoulos, Nikos Bezirgiannidis, Ioannis Komnios, Ioannis Psaras, Sergi Rene, Adisorn Lertsinsrubtavee, Arjuna Sathiaseelan, Susana Perez Sanchez, Francisco Almeida, Francesco Amorosa, Giammichele Russi, Angela d'Angelo, Alberto, "D2.3 System and Network Requirements Specification -final version-," UMOBILE Project, July 2017.

[5] Sotiris Diamantopoulos, Christos-Alexandros Sarros, Vassilis Tsaoussidis, Sergi Rene, Ioannis Psaras, Jon Crowcroft, Adisorn Lertsinsrubtavee, Carlos Molina-Jimenez, Paulo Mendes, Iñigo Sedano, Susana Sanchez Perez, Rute Sofia, "D3.2 UMOBILE architecture report (2)", UMOBILE Project, July 2017

[6] Sergi Rene, Ioannis Psaras, Sotiris Diamantopoulos, Christos-Alexandros Sarros, Vassilis Tsaoussidis, Jon Crowcroft, Adisorn Lertsinsrubtavee, Carlos Molina-Jimenez, Paulo Mendes, Iñigo Sedano, Susana Sanchez Perez, Rute Sofia, "D3.4 UMOBILE ICN layer abstraction – final specification", UMOBILE Project, July 2017

[7] Angela d'Angelo, Gianmichele Russi, Francesco Amorosa, Alberto Pineda, Jose Pablo Salvador, Inigo Sedano Perez, Sergi Rene, Ioannis Psaras, Sotiris Diamantopoulos, Christos-Alexandros Sarros, Vassilis Tsaoussidis, Adisorn Lertsinsrubtavee, Carlos Molina-Jimenez, Paulo Mendes, Seweryn Dynerowicz, Omar Aponte, Rute Sofia, "D5.3 Proof of Concept (1)", UMOBILE Project, January 2017

[8] R. Adams, "Active queue management: A survey," IEEE Communications Surveys Tutorials, 2012.

[9] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in IEEE INFOCOM, 2001.

[10] K. K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," ACM Trans. Comput. Syst., vol. 8, pp. 158–181, May 1990.

[11] B. Ford and J. Iyengar, "Breaking up the transport logjam," in ACM HotNets-VII, 2008.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

[12] B. Ford and J. Iyengar, "Efficient cross-layer negotiation," in ACM HotNets-VIII, 2009.

[13] P. P. Mishra and H. Kanakia, "A hop by hop rate-based congestion control scheme," in ACM SIGCOMM, 1992.

[14] S. Sinha, S. Kandula, and D. Katabi, "Harnessing TCP's burstiness with flowlet switching," in ACM HotNets-III, 2004.

[15] V. Jacobson, "Congestion avoidance and control," in ACM SIGCOMM, 1988.

[16] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," Computer Networks and ISDN Systems, vol. 17, no. 1, pp. 1 – 14, 1989.

[17] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," eprint arXiv:cs/9809099, Sept. 1998.

[18] I. Psaras, L. Saino, and G. Pavlou, "Revisiting resource pooling: The case for in-network resource sharing," in Proceedings of the 13th ACM Workshop on Hot Topics in Networks, HotNets-XIII, (New York, NY, USA), pp. 24:1–24:7, ACM, 2014.

[19] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the internet," in IWQoS'05, pp. 271–285.

[20] C. M. D. Pazos and M. Gerla, "A rate based back-pressure flow control for the internet," in IFIP HPN, 1998.

[21] S. Sarkar and L. Tassiulas, "Back pressure based multicast scheduling for fair bandwidth allocation," in IEEE INFOCOM, vol. 2, pp. 1123–1132 vol.2, 2001.

[22] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," IEEE/ACM Trans. Netw., vol. 12, pp. 2–16, Feb. 2004.

[23] L. Saino, C. Cocora, G. Pavlou, CCTCP: a scalable receiver-driven congestion control protocol for content centric networking, Proceeding of IEEE ICC'13, 2013.

[24] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. claffy, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, Named data networking, ACM SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 66–73.

[25] I. Moiseenko and D. Oran. Flow Classification in Information Centric Networking draft-moiseenko-icnrg-flowclass-00, July 22, 2016. Icnrg Internet-Draft, Intended status: Informal, Expires Jan 23, 2017.

[26] C. Xia and M. Xu. RRCP. A Receiver-Driven and Router-Feedback Congestion Control Protocol for ICN. In Proc. Third Int'l Conf. on Networking and Distributed Computing, 21-24 Oct, Hangzhou, Zhejiang, China 2012.

[27] D. Katabi, M. Handley and C. Rohrs, Congestion Control for High Bandwidth-Delay Product Networks, In Proc. SIGCOMM'02, August 19-23, Pittsburgh, Pennsylvania, USA, 2002.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124

[28] Y. Ren, J. Li, S. Shi, L. Li, G. Wang and B. Zhang. Congestion Control in Named Data Networking – A survey. Computer Communication, Vol. 86, July, pag. 1–11, 2016.

[29] G. Carofiglio, M. Gallo, and L. Muscariello. ICP: Design and evaluation of an interest control protocol for Content-Centric Networking. In IEEE INFOCOM NOMEN, pages 304–309, 2012.

[30] G. Carofiglio, M. Gallo, and L. Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In ACM SIGCOMM ICN, 2012.

[31] T. Janaszka, D. Bursztynowski, and M. Dzida. On popularity-based load balancing in content networks. In ITC-24, 2012.

[32] S. Oueslati, J. Roberts, and N. Sbihi. Flow-aware traffic control for a content-centric network. In IEEE INFOCOM, 2012.

[33] N. Rozhnova and S. Fdida. An effective hop-by-hop interest shaping mechanism for CCN communications. In IEEE INFOCOM NOMEN, pages 322–327, 2012.

[34] S. Salsano et al. Transport-layer issues in information centric networks. In ACM SIGCOMM ICN, 2012.

[35] C. Yi et al. A case for stateful forwarding plane. Comput. Commun., 36(7):779–791, Apr. 2013.

[36] P. P. Mishra and H. Kanakia. A hop by hop rate-based congestion control scheme. In ACM SIGCOMM, 1992.

[37] Klaus Schneider, Cheng Yi, Beichuan Zhang, and Lixia Zhang. 2016. A Practical Congestion Control Scheme for Named Data Networking. In Proceedings of the 3rd ACM Conference on Information-Centric Networking (ACM-ICN '16). ACM, New York, NY, USA, 21-30. DOI: http://dx.doi.org/10.1145/2984356.2984369

[38] A. Afanasyev, I. Moiseenko, L. Zhang, ndnSIM: NDN simulator for NS-3, Tech. Rep. NDN-0005, NDN Project (July 2012).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645124