Action full title:

Universal, mobile-centric and opportunistic communications architecture

Action acronym:

UMOBILE

Deliverable:

D4.1 “Flowlet Congestion Control – Initial 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, Democritus University of Thrace

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	D4.1 Flowlet Congestion Control – Initial Report
WP Number	WP4
WP Leader	SENCEPTION
Task Leader (s)	UCAM
Authors	**DUTH:** Sotiris Diamantopoulos, Ioannis Komnios, Vassilis Tsaoussidis **UCL:** Ioannis Psaras, Sergi Rene **UCAM:** Jon Crowcroft, Adisorn Lertsinsrubtavee, Arjuna Sathiaseelan, Carlos Molina-Jimenez
Contact	i.psaras@ucl.ac.uk, s.rene@ucl.ac.uk
Due date	M18: 31/07/2016
Actual date of submission	--/--/2016

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	07/07/16	First draft of the flowlet congestion control
Version 0.2	19/07/16	Second draft to the consortium of the flowlet congestion control
Version 0.3	30/07/16	Some modifications before the final version
Version 0.4		

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

Table of Contents

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 traffic refers to outgoing data such as short message, photo or uploading video clips that are sent from user equipment.
Downstream	Downstream traffic refers to data is obtained by use equipment from network. This includes downloading files, web page, receiving messages, etc.

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.
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.
Application Service	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. 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.
NDN	Named Data Networking

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 (D4.1) is the first out of five to be produced by WP4 to 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 interactions between two different domains (see Figure 1):

- The *UMOBILE Domain* which is a local network running the UMOBILE protocols (e.g., NDN, DTN). Providing the Internet access in this domain is optional as UMOBILE users can access the local services through public hotspot (e.g., WiFi AP) or WiFi direct (e.g., direct connection among users).
- The *Internet Domain* is the conventional TCP/IP Internet and is an extension of the UMOBILE Domain where UMOBILE users (running either TCP/IP, NDN or DTN protocols) can access Internet services. For instance, as shown in the figure, the UMOBILE network can have a proxy or gateway running TCP/IP in order to connect to the Internet.

D4.1 discusses the first version of a congestion control protocol, called In-Network Resource Pooling Protocol (INRPP) that addresses congestion control in TCP/IP networks. It extensively covers the specifications for a Flowlet Congestion Control to be used in the Internet Domain. In addition, we introduce a preliminary discussion of the development of the Flowlet Congestion Control to be used in the UMOBILE Domain. The final version of the protocol will be discussed in D4.2.

In D4.1 we also provides a broad performance evaluation through simulations to assess the INRPP mechanisms and to quantify the improvement of INRPP over other congestion control protocols.

D4.1 is based on deliverables D2.1, D2.2, D3.1 and D3.3. Deliverables D2.1 and D2.2 describe the requirements of the end-user and the system, respectively. On the other hand, D3.1 and D3.3 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 detail a set of mechanisms required for the INRPP to implement the new congestion control over TCP/IP networks
- We provide an extensive evaluation of the INRPP congestion control approach
- We describe the next steps for an implementation of the INRPP congestion control over NDN-based networks that can be extended to the UMOBILE Domain.

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 D3.3) we devise two different domains (see Figure 1), the Interned Domain, where UMOBILE users can access Internet-based services through public-hotspots or network gateways, and the UMOBILE Domain, where the communications are inherently wireless---this is why we call it the UMOBILE wireless Domain in this document. Notice that well planned infrastructure communications in the UMOBILE wireless Domain is optional, thus in disaster-affected or neglected rural areas, UMOBILE users are still able to transmit their data to other users. In this document, we first focus on a flowlet congestion control based on in-network resource pooling for the Internet Domain.

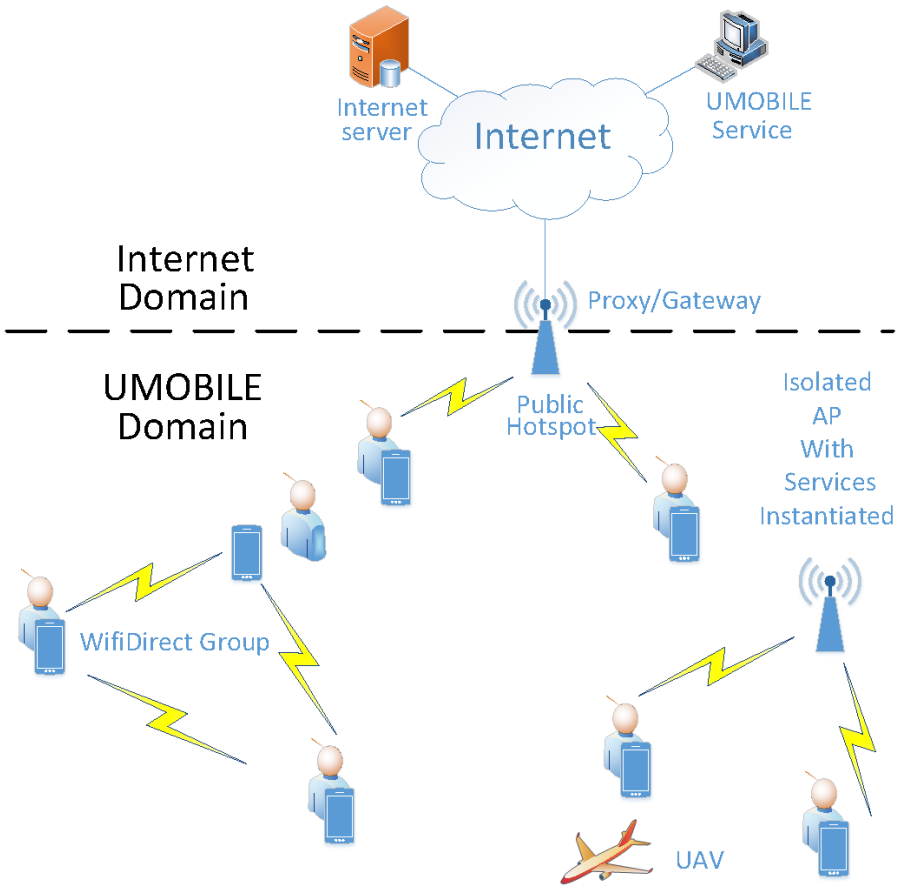


Figure 1 - UMOBILE architecture

Congestion control refers to techniques and mechanisms that can avoid congestive collapse between two end points by controlling the rate of sending packet 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, and ii) the demand factor: there might be excessive demand for bandwidth over some particular area/link. The Transport Control Protocol (TCP), for instance, defends against the input load factor through the

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

Additive Increase/Multiplicative Decrease transmission model [8,9], 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.

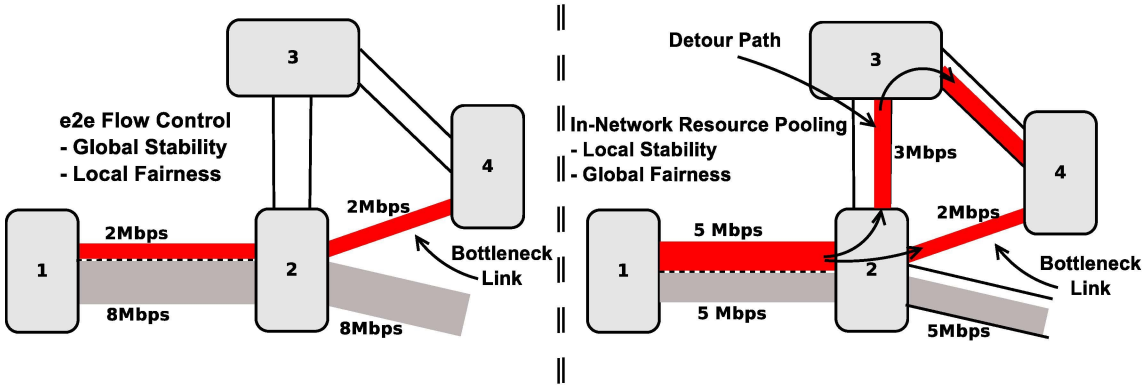


Figure 2 - Left: e2e Flow Control: Bandwidth is split according to the slowest link on the path. Right: In-network resource pooling: Bandwidth is split equally up to the bottleneck link (global fairness). Detour applies to guarantee local stability.

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 in-network resource pooling propose that: i) stability should be local, and ii) fairness should be global. Local stability demands that the node before the bottleneck link takes 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 Fig. 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 [11], given by, 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) [10], 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

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

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 modes of operation: i) push: content is pushed as far in the path as possible in an open-loop, processor sharing manner [11], 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 enters a backpressure mode of operation [12, 13] to avoid overflowing of the cache. During the backpressure mode, the nodes enter a closed-loop mode, where an upstream node sends one data packet per one received ACK to the backpressuring downstream node.

In this first 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 extensive performance evaluation of the INRPP congestion framework in the Internet domain over TCP/IP. However, we also include a section (Section 3) about the next steps to follow to implement a flowlet congestion control based on INRPP in the UMOBILE Domain (over NDN), that will be completed and detailed in the second version of this document that we will submit in month 30.

The document is organized as follows.

- **Section 2** presents an overview of congestion control in UMOBILE and related work in TCP/IP and NDN networks.
- **Section 3** presents the framework overview and the mechanisms of the INRPP for congestion control over TCP/IP.
- **Section 4** introduces the next steps to implement INRPP over an NDN-based solution for the UMOBILE project.
- **Section 5** concludes this deliverable.

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 UMOBILE

As mentioned in the Introduction section, congestion control has been intensively studied in the conventional TCP protocols for which several algorithms are known to ameliorate the problem. The techniques used heavily depend on the particularities (for example, the existence of unique source and destination addresses) of TCP/IP, consequently, they cannot be directly ported to ICN networks where the problem has received less attention.

To help cover the gap, in D4.1 we introduce the Flowlet Congestion Control, which to our knowledge, is one of the first attempts to address the problem. In this section, we will include a discussion of other alternatives that have been brought to our attention. To place the Flowlet Congestion Control in context, we include a discussion of congestion control in TCP/IP.

2.1 Congestion control in TCP/IP

Regarding the Internet domain, the common practice among ISPs to move the bottleneck to the edge of the network (i.e., DSLAM to user) restricts users from taking up as much bandwidth as possible. As we move towards a FTTx environment, however, the bottleneck will inevitably move to the core of the network causing severe congestion events. Over-provisioning of core links will not be an option anymore and therefore alternative solutions will need to be found. Multi-Path TCP [40] has received wide attention recently due to its ability to take advantage of multiple e2e paths. However, the requirement for multihoming of MPTCP (and mTCP [41]) have driven adoption of MPTCP to controlled, data-centre environments mainly [42]. INRPP builds on the previously proposed Resource Pooling Principle [43, 44] and extends it to also utilise midpath multipath without the equal-cost requirement of ECMP [45].

Multipath routing on the other hand has been studied in the context of traffic engineering in the core of the network [46-49], mainly for load-balancing reasons [45,49]. Despite extensive studies on multi-path routing [50-52] and multipath congestion control [53-55], these two arguably complementary areas remain remarkably decoupled. There has been no previous attempt to combine the benefits of multipath routing and congestion control into a common resource pooling principle in order to improve overall resource utilisation.

In the context of the H2020 POINT (iP Over IcN– the betTer IP) project, the partners involved recently proposed a new Multiflow Congestion Control with Network Assistance [56]. This new congestion control is a novel hybrid congestion control algorithm. It is based on MPTCP and uses an in-network module that offers essential topological information to avoid sharing bottlenecks between different subflows. As a result, it extends the bandwidth available to the end-users. However, this Multiflow Congestion Control with Network Assistance relies on MPTCP. Thus is suffers from the same drawbacks as TCP regarding the aforementioned e2e flow control. Even when not sharing bottlenecks, MPTCP has not enough knowledge on the network to provide global fairness and local stability due to e2e flow control.

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

With the In-Network Resource Pooling Protocol we make a first attempt to bring the worlds of multipath congestion control and multipath routing closer together. Although much of INRPP's mechanisms can be replaced or redesigned to fit to specific network needs, our detailed design and evaluation shows that the proposed set of mechanisms work smoothly together. At the same time, the significant performance gains observed motivated as to research a combined multipath routing and congestion control framework.

2.2 Congestion control in ICN

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 [34]. 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 [35]. 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 D4.1 is the discussion of Flow Classification in ICN presented in an informal Internet draft [36] 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

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

of the name prefixes of the Interest and Data packets exchanged between consumers and producers: equivalent classes of flow are associated to the names in the corresponding Interest and Data packets. In [37] the authors discuss RRCP--- a congestion control protocol for designed specifically for ICN networks and based on the XCP (eXplicit Control Protocol) presented in [38]. Though the authors show only simulation results, they argue that their solution is promising as it takes advantage of the 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 [39].

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., [24-30]). Closer to our INRPP work are [27], [28] and [30]. Although the protocol in [30] 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 [30] takes place at the Interest phase (instead of the data phase), hence, its accuracy is bound to be outdated by approximately. The work in [28] on the other hand, is based on hop-by-hop rates to shape the rate of interests and therefore, data as well (similarly to [31]). 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 TCP/IP networks

Congestion control has been 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 [1]. 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., [2,3]). Solutions to these problems [4, 5, 6, 7] have continuously and stubbornly been rejected mainly due to fear of instability.

In this Section we present a new congestion control for TCP/IP networks based in In-network resource pooling, that we named In-network Resource Pooling Protocol (INRPP). 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 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 [14]) for nine ISPs (see Table 1). Indeed, we found 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.

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

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

INRPP requires several new mechanisms and protocols to be implemented by the routers, while minor modifications are required for the TCP/IP agents at the end-points, more substantial changes are necessary in the routers. Next we describe an overview of the framework, we discuss how backpressure requests propagate, and we explain the cache storage system. Then, we discuss computation of detour paths and the computation/advertisement of residual bandwidth on detour paths. Finally, we discuss the changes to the TCP agents at the end-points.

3.1 Framework Overview

An INRPP router has a single cache storage, which is shared among all interfaces; that is, excess data from all the bottleneck links are stored in the cache. An INRPP router is shown in Figure 3. For the purposes of this report we assume that flows follow 1-hop detours only, i.e., no 2- or more-hop detours are considered. Each out-going link of a router is associated with zero or more one-hop detour paths, which are computed through a minor extension to the existing distributed path computation mechanism of the intra-domain protocols (see Section 3.3).

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

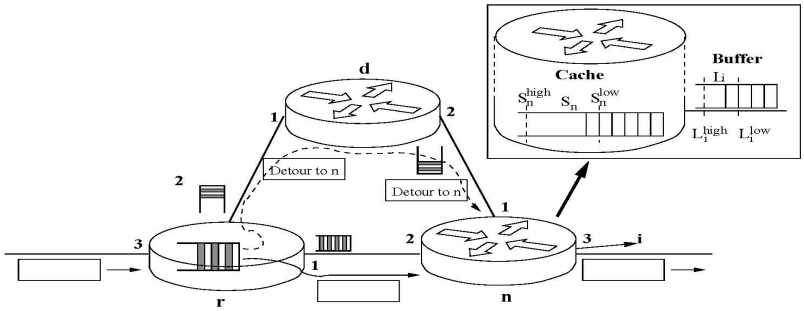


Figure 3 - INRPP router (zoomed in the box on the right) and a detour example: Router r has a detour path through router d to reach n as shown with the dashed lines

At any given time, each interface of an INRPP router is in one of the four states: *Push (P), Store and Detour (S&D), Backpressure (B), and Disable Backpressure (DB)* state as shown in Table 2. Below, we describe each of the states using a state transition diagram depicted in Figure 4. Our notation is given in Table 3:

State Abbreviation	Definition
P	Push
S&D	Store and Detour
B	Backpressure
DB	Disable Backpressure

Table 2 - INRPP states

- *Push (P)*: In this state, the router interface is able to forward all the incoming data to its outgoing interface in a processor sharing manner, without using its cache storage or detour paths. The Push state is maintained while there is no congestion, i.e., no excess data, at the outgoing interface queue. In order to detect congestion at its interfaces, routers continuously monitor the occupancy of their outgoing interface buffers. We define an upper-bound and a lower-bound on the occupancy of the interface queues to detect congested and uncongested state at the queues, respectively. In particular, if the buffer occupancy of an interface of a router exceeds (i.e., upper bound), then interface moves to *Store-and-Detour (S&D)* mode.

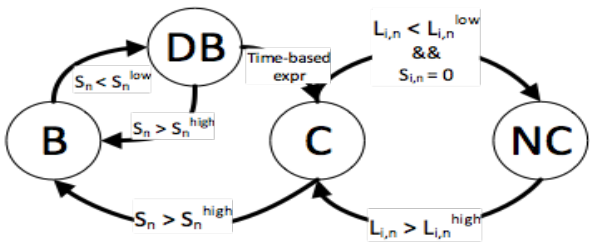


Figure 4 - INRPP state transition diagram

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

- *Store-and-Detour (S&D)*: While its interface is in *S&D* state, the router starts storing all the incoming packets, whose next-hop is determined (by the forwarding table lookup) to be; we refer to such packets as the interface's traffic, denoted with the symbol (see Table 3). After this point, pulls packets directly from the cache and sends them out at the maximum rate of its outgoing link, denoted. At the same time, the routers continuously monitor their share of residual capacity in all of their detour paths through single-hop advertisements. If there are detour paths with non-zero residual bandwidth capacity, the router pulls packets from the cache and sends them through the detour interfaces (that is, the interfaces facing the first link of the detour paths) at the rate equal to the total residual bandwidth of all the detour paths. The total residual bandwidth available for node's interface on all its detour paths is denoted as; we provide a detailed explanation of the residual bandwidth calculation in Section 3.3.

Symbol	Definition
$L_{i,n}$	Queue occupancy at interface i of node n
$L_{i,n}^{high}$	Upper-bound occupancy at the buffer of interface i at node n
$L_{i,n}^{low}$	Lower-bound occupancy at the buffer of interface i at node n
S_n	Cache occupancy at node n
S_n^{high}	Upper-bound occupancy of the cache at node n
S_n^{low}	Lower-bound occupancy of the cache at node n
$C_{i,n}$	Link i capacity at node n
$T_{i,n}$	Link i traffic at node n
$R_{i,n}$	Residual bandwidth on the detour paths for link i at node n

Table 3 - INRPP design notation

The interface i stays in the *S&D* state while excess data for i is being forwarded and the total cache occupancy of the router n is below the upper-bound S_n^{high}. When the arrival rate of the interface i's traffic stays below the rate at which i's traffic is sent out through the detour paths and the interface itself, i.e.,

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

$T_{i,n} < C_{i,n} + R_{i,n}$, the amount of cache storage used by i's traffic eventually drops to zero. Note that it is possible for a router to have some interfaces congested while others are not. If the occupancy of the main cache increases beyond the upper-bound S_n^{high}, and if the interface i currently has traffic stored in the cache, then i switches to *Backpressure (B)* state; the latter condition is denoted: $S_{i,n} > 0$ in Figure 4. When $S_{i,n}$ reaches zero and the occupancy of the queue buffer $L_{i,n}$ falls below the threshold $L_{i,n}^{low}$, the interface is no longer congested and thus, switches back to the Push mode.

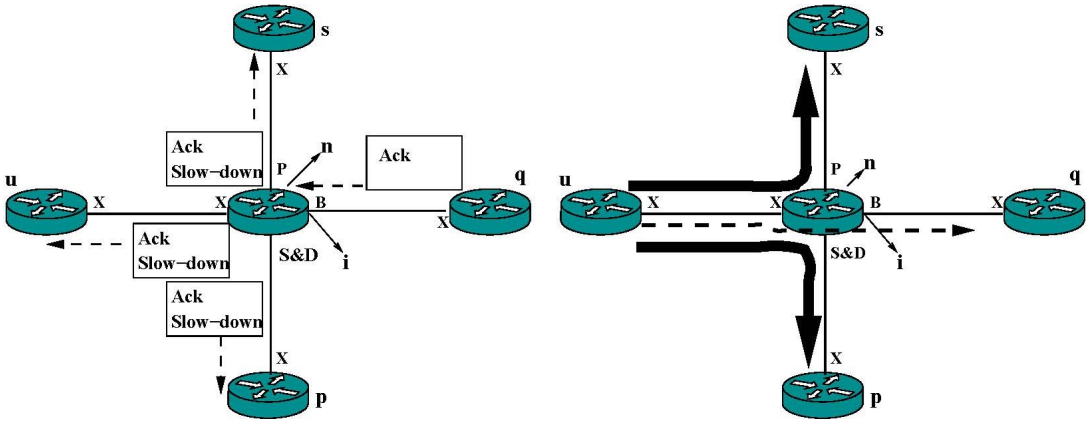


Figure 5 - Backpressure Example: On the left-side is the placing of slow-down notifications in ack packets and on the right-side is the slowing-down of traffic shown with dashed line

- *Backpressure (B)*: The system enters a closed-loop, backpressure mode of operation in order to reduce the occupancy of the cache. In particular, when the occupancy of the router n's cache goes beyond the upperbound S_n^{high}, all the interfaces, which have traffic in the cache, i.e., all i s.t. $S_{i,n} > 0$, switch from *S&D* mode to *Backpressure* mode of operation. During backpressure mode, an interface i sends "slow-down" notifications to upstream nodes. Upon receiving the slow-down request, the upstream nodes of n start performing ack pacing operation on only the flows which are heading towards interface i.

  As an example, consider Figure 5 showing the before (left) and after (right) snapshots of a topology, where node n (in the centre) is receiving traffic from an upstream node u and relaying the traffic to downstream nodes s, q and p. The state of each interface is given in upper-case letter next to the out-going interface. The interface of n connecting to q, which is named i is in *Backpressure* state. The state "X" is assigned to some interfaces as a "don't care state", which means the actual state of the interface is irrelevant. When u receives a slow-down notice from n, it starts performing ack pacing on only the

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

flows that are part of the traffic from $u \rightarrow n \rightarrow q$ (shown with the dashed line on right hand side). Flows to any other interface of n will not be affected. This is achieved through nonce messages propagated upwards, which we discuss next.

The slow-down notifications sent from i are piggybacked in the acknowledgement packets arriving to i from downstream. On the left of the Figure 5, interface i places the slow-down request in the acknowledgement packets arriving from downstream node q. In the acknowledgement packets, i inserts a nonce, which is effectively an alias name, i.e., fixed length bit string, for i. The upstream nodes store the nonce and forward the packets with the nonce further upstream towards the end-point.

The acknowledgement packets eventually reach the sender-side TCP agent (that is, the originator of the corresponding data packet). Once the TCP agent at the sender side observes the slow-down request in the ack packets, it starts placing the nonce in its outgoing data packets. When a node receives a data packet with a nonce, it compares the nonce with all the locally stored nonces. In case of a match, the packet is stored in the cache regardless of the state of the interface that the packet is to be forwarded to. An upstream node, e.g., node u in the previous example, sends one data packet with nonce, for each arriving acknowledgement packet with the same nonce. This effectively slows-down the flow of traffic going downstream to a backpressuring interface in the downstream node. Once the occupancy of n's cache drops below S_n^{low}, interface i leaves the *Backpressure* state and switches to *Disable Backpressure* state.

- *Disable Backpressure (DB)*: Before an interface i in a *Backpressure* state can switch back to *S&D*, i first enters a transition state named *Disable Backpressure (DB)*. During this state, n's interface i sends a "cancel" notice along with the nonce corresponding to i to upstream nodes of n who have been caching traffic heading downstream to i. The cancellation notification is again piggybacked in the acknowledgement packets, i.e., packets arriving to i from downstream. The upstream nodes of n receiving the cancellation notice, erase the nonce from their list of locally stored nonce list and stop caching packets heading downstream to n which carry the nonce. The *DB* state is maintained for a fixed duration, and the state is eventually changed to *S&D*. The duration is set to approximately a second in the implementation to ensure all or most of the upstream (both direct and indirect) nodes receive the cancellation notification. However, as we explain in Section 3.1, the state maintained for each stored nonce also expires after a duration of inactivity, i.e., no ack arrivals containing the nonce, to make sure nonces are erased. If during the *DB* state, the

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

occupancy of the cache exceeds the upper-bound, then the interface switches back to Backpressure state.

For the backpressure mechanism using piggybacked (slow-down, cancel) upstream notifications to work, INRPP flows require symmetric paths, where data and the corresponding ack packets traverse the same paths in the reverse direction. However, INRPPP could be adjusted to work without symmetric paths, if the data packets were to be used for piggybacking notifications, instead of ack packets: the nonce and the notification would travel to the receivers in the tagged data packets and from the receivers back to the senders in the ack packets, and finally the senders would eventually place the nonce and the instruction in their next data packets. This change would add an additional RTT for the routers to receive the notifications, and an additional mechanism would be needed for the notifications to be ignored or not seen by the downstream nodes of the backpressuring interface. We leave a detailed investigation for the possible consequences of asymmetric paths for future work.

An important consideration we discuss next is the extent to which the slow-down operation is propagated in the upstream nodes of a backpressuring node. Triggered by slow-down notifications, nodes enter a closed-loop mode of operation and transfer one data packet for each ack received. However, the closed-loop operation is performed hop-by-hop, and initially it is performed only between the immediate up- stream node and the node with the backpressuring interface, e.g., nodes u and n in Figure 5. Extending the closed-loop operation to more than one-hop may be required if the upstream node's cache is also nearly full. Next, we discuss how upstream nodes process notifications arriving from downstream.

3.2 INRPP mechanisms

INRPP requires several new mechanisms to be implemented by the routers, while minor modifications are required for the TCP/IP agents at the end-points. In this section, we provide implementation details of the several mechanisms comprising the In-Network Resource Pooling Protocol. We discuss how backpressure requests propagate and we explain the cache storage system. Then, we discuss computation of detour paths and the computation/advertisement of residual bandwidth on detour paths. Finally, we discuss the changes required at the TCP agents at the end-points.

Processing of downstream requests

As discussed above, arrival of a slow-down notification to a node u from the downstream node n initiates a closed-loop operation between u and n. During this operation, router u caches packets carrying the nonce —an alias name for the interface of a downstream node— received in the slow-down request. Router u sends one data packet from the cache for each ack packet with matching nonces. It is not clear, however, whether u should request closed-loop operation from its own upstream. Our proposal is for u to not

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

propagate the slow-down request further upstream as long as u has sufficient cache space and therefore, flows can take advantage of the open-loop operation until the packets hit a bottleneck node with nearly full cache.

The pseudocode in Algorithm 1 shows the processing of ack packets at an interface. Depending on its node's current "mode of operation", the interface either propagates an incoming slow-down instruction up-stream or clears the notification field and only propagates the nonce to prevent the slow-down notification to propagate further. Remember that the nonce in the ack packets is needed by the senders to tag its future packets, and therefore it is kept in the ack packets by each router. Each node is either in CLEAR (CLR) or PROPAGATE (PR) state at any point in time and it is initially set to CLR at each router. If the router's cache occupancy exceeds the upper-bound, the interface switches to PR mode (Line 3), and stays in this mode for as long as the occupancy of the cache is above the lower-bound (see Algorithm 1). In the PR mode, the slow-down notifications are propagated upstream. On the other hand, if the interface is in CLR mode, the slow-down notifications are cleared from the ack packets (Line 11) before forwarding the ack upstream.

The nonce, which comes along with the slow-down notification is added to the list of nonces stored by the router (Line 8). Upon receiving an ack packet with slow-down notification, the interface fetches from the cache a data packet, which belongs to the same flow as the ack packet, and forwards the data packet (Line 13, 14). If, on the other hand, a nonce with a cancel notification is received, the nonce is removed from the list of nonces stored by the router (Line 16). In addition to the cancellation mechanism, nonces are kept in the router as a soft state, which means they eventually expire if the router does not receive any packets containing the nonces for a period of time. In Algorithm 1, a timer is set (or reset) to expire for each nonce upon receiving a slow-down request (Line 16) to implement soft state.

Cache storage system

INRPP uses in-network storage in order 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 flows while performing ack pacing, ii) forward packets ordered by sequence (of arrival) to avoid reordering is- sues at the receiver2, but most importantly, iii) avoid head of line blocking at the cache.

INRPP caches are indexed by: i) the primary output interface, that is, not the detour interface, if any, and ii) the flow id. As flow id, we use the hash of the 5-tuple <source address, destination address, source port, destination port, protocol>.

Outgoing interfaces pull packets from the cache at the rate determined by the incoming rate of ack packets when the ack packet contains slow-down notification. In that case, the

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

router fetches a data packet from the cache whose flow identifier matches the ack packets. Next, we discuss how INRPP nodes discover the available residual bandwidth on detour paths.

Detour path information

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.

Consider the example topology in Fig. 3 where a router r has a one-hop detour path through d to reach n. When r's interface facing n (labeled 1) is congested, the interface switches to *S&D* mode. At this point, r starts caching excess data and starts demultiplexing packets between the primary outgoing interface and the detour interface(s) as shown in Fig. 3. A router sends cached traffic through a detour path only if the path has residual, i.e., unused, bandwidth to forward the traffic. A detour path has residual bandwidth if it operates in the Push mode (see Section 2). In the example of Figure 3, r uses the detour path, shown with the dashed line, only if interface 2 of r and interface 2 of d are both in *Push* mode.

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 Fig. 3, 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 report. Routers then periodically advertise the residual bandwidth on their interfaces, which are in Push mode to their immediate neighbours.

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

Algorithm 1 Processing of acknowledgement packets

```
1: function ACK PROCESSING(Mode m, Ack Packet p)
2:     if S_n > S_n^high then
3:         m ← PROPAGATE
4:     else if S_n < S_n^low then
5:         m ← CLEAR
6:     end if
7:     if p.notification ==SLOW-DOWN then
8:         L_nonces.add(p.nonce)        ▷ Store the nonce
9:         set_expiration_timer(p.nonce)
10:        if m == CLEAR then
11:            clear(p.notification)
12:        end if
13:        Data Packet d ← Cache.get(p.flow)
14:        forward(d)
15:    else if p.notification ==CANCEL then
16:        L_nonces.remove(p.nonce) ▷ Delete the nonce
17:    end if
18:    forward(p)
19:    return s
20: end function
```

In order to make sure the detoured traffic eventually reaches the intended next-hop, routers tag their detoured packets with their original next-hop IP address. For example, in the previous example router r tags its detour traffic going to d with its original next hop n 's address as shown in Fig. 3. Tagging the detour traffic is also useful in order to distinguish detour traffic from regular traffic, which in turn, allows routers to keep track of detour traffic per neighbour node.

In the previous example, d advertises the amount of residual capacity on the link of its interface labelled 2 to router r, denoted as $R_{2,d}$. Then, r knowing the residual capacity on its own interface 2, denoted as $R_{2,r}$, can determine the residual bandwidth on the detour path from $r \rightarrow d \rightarrow n$ as: min($R_{2,r}$, $R_{2,d}$). Upon computing its detour paths, each router forms a detour interface lookup table to map each next-hop interface to its set of detour interfaces. As an example, router r's lookup table is shown in Table 4, where interfaces 1 and 2 detour traffic for each other and 3 has no detour interfaces.

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

Primary Outgoing Interface	Detour Interface(s)
1	2
2	1
3	Null

Table 4 - Detour interface lookup (router *r*)

One complicating factor when advertising residual bandwidth of a link is when the link is used by multiple detour paths. As an example, consider the topology in Fig. 6 where four routers: p, q, r, and s are interconnected. Routers q and r are both directly connected to s with direct links, and they both have a 1-hop detour path through p to reach s as shown with dashed lines. Router p on the other hand, has two detour paths to reach s: $p \rightarrow q \rightarrow s$ and $p \rightarrow r \rightarrow s$.

Routers iteratively revise the amount of bandwidth advertised to each neighbor to reach an approximately fair allocation of bandwidth resources between the detour paths. Based on information from the link-state protocol discussed above, router p discovers that its link to s is shared by detour traffic from q and r heading to s shown as dashed lines in Figure 6. The residual bandwidth $R_{i,n}$ on the interface i of a node n is computed simply as $R_{i,n} = C_{i,n} - T_{i,n}$, where $C_{i,n}$ is the bandwidth capacity and $T_{i,n}$ is the amount of total traffic flowing through interface i.

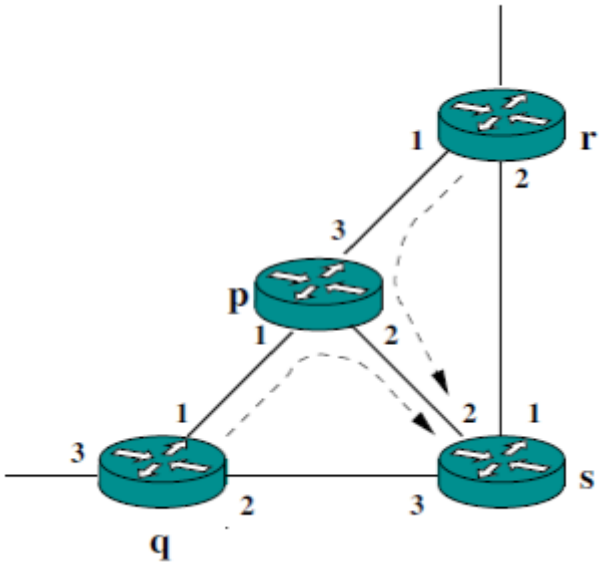


Figure 6 - Detour Example: The routers q and r use detour paths through router p to reach s as shown with the dashed lines

Instead of simply splitting the residual capacity between nodes q and r, router p allocates the residual bandwidth of interface 2, $R_{2,p}$, to nodes q and r in a max-min fair manner.

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

We use a slightly modified version of the original max-min fairness algorithm, which is based on the actually used traffic by each of the neighbour routers. The ultimate purpose is to avoid underutilising the residual capacity available through node p, e.g., in case r has lower demand than advertised by p and q has higher demand (or vice versa). This is done through monitoring of the incoming traffic from both nodes q and r and adjusting the advertised residual bandwidth accordingly.

The detailed steps for this operation are given in Algorithm 2. In this operation, it is essential for router p to be able to distinguish between regular and detour traffic. In our case, this is done through the packet tags attached to detour packets, as discussed earlier. Router p having two detour paths using its link on interface 2, can simply advertise $R_{2,p}$, but this could lead to receiving more detour traffic than the residual bandwidth. Alternatively, router p can share the residual bandwidth equally by advertising half of $R_{2,p}$ to both neighbours. However, the problem in this case is the possibility of underutilising the residual resources since r may have less than $R_{2,p}$ to send, while q might have higher demand than $R_{2,p}$. In order to compute an approximately fair allocation, each router monitors the amount of detour traffic sent from each neighbour to observe if the neighbour is using the advertised amount. This requires routers to be able to distinguish detour traffic from other traffic, and they do by checking the packet for a tag; as explained above, routers tag their detour traffic with the address of the next-hop. The computation of the advertised residual bandwidth amount sent to a node n when m nodes are competing for the residual bandwidth is given in Algorithm 2. Advertisements are sent periodically in the time-scales of milliseconds, and each subsequent advertisement revises the bandwidth advertised to each node to reach an approximately fair allocation. Initially, a router advertises an equal share of the unused bandwidth on its link to all neighbors, which potentially use the link for detouring traffic. Then, after observing the moving average of the actual traffic sent until the next iteration, the router advertises an amount based on the actual used amount. In particular, each router is advertised an amount equal to the actual amount of traffic it sent plus an equal share of the current residual bandwidth (Line 8). If there is no residual bandwidth available, the router advertises an equal share of the total bandwidth used by all the detour traffic, which is computed in Lines 3- 5.

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

Algorithm 2 Computation of residual bandwidth on interface i of node p advertised to node n when m nodes are competing for the residual bandwidth

```
1: function ResidualUpdate(node n, int m)
2:     DetourTraffic ← 0
3:     for each neighbor k do
4:         DetourTraffic = DetourTraffic + DetourTrafficFrom(k)
5:     end for
6:     R_{i,p} ← C_{i,p} − T_{i,p}
7:     if R_{i,p} > 0 then
8:         return detourTrafficFrom(n) + R_{i,p}/m
9:     else                ▷ No residual bandwidth available
10:        return DetourTraffic/m
11:    end if
12: end function
```

INRPP sender/receiver

In order to comply with the above-mentioned mechanisms, we use a modified, rate-based version of TCP. Our resulting INRPP end-point client complies with the feedback signals received from notifications (e.g., slow-down) in the ACK packets, as presented in Algorithm 1. The fast-recovery mechanism at the sender is disabled; however, the timeout-based retransmission mechanism is kept: even though INRPP routers no longer experience congestion-related packet drops, as opposed to end-to-end congestion protocols, packets need to be re- transmitted in the case of packet corruption due to channel interference.

The INRPP sender initially works in an open-loop manner and forwards packets according to processor sharing. When the INRPP sender receives a nonce within an ACK packet, it stores the nonce and copies it to the future data packets of this flow. The nonces as well as the slow-down and cancel notifications are stored within INRPP's packet option fields. When an INRPP sender receives a slow-down request, it adapts the sending rate of the specific flow to the rate of acknowledgement packets (i.e., closed loop operation based on the "one-out, one-in" principle). On the other hand, when an INRPP sender receives a "cancel" notification through an ACK packet, it switches back to open-loop mode and sends again at the maximum possible rate available through its outgoing interface.

The INRPP receiver does not require any modifications compared to a standard TCP receiver. It is worth noting, however, that INRPP receivers might see out-of-order packets, due to reordering. Although reordering might happen due to detouring of some parts of the flow, detour paths add very small extra delays (in the order of a few milliseconds in our extensive evaluation), given they are only 1-hop detours. Also, note that INRPP avoids packet loss and lengthy retransmissions, which normally add substantial delay and cause

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

Head-of-Line (HoL) blocking. That said, reordering in our case differs from the more complicated cases where reordered packets experience substantially longer delays.

3.3 INRPP results

We next engage in a detailed investigation of the performance of the proposed INRPP scheme in an infrastructure scenario. In the following sections, we compare the performance of INRPP with TCP, RCP, and MPTCP for various topologies and scenarios. We implemented INRPP in ns-3 [15], ported the existing ns-2 implementation of MPTCP to ns-3 and used RCP's existing implementation for ns-3 [16]. We use the New Reno version of TCP.

We evaluated INRPP in a three different scenarios. The first one is a very simple topology (dumbbell topology), in order to show in detail the operation of the different INRPP mechanisms detailed in Section 2.2. The second scenario is a simple multihomed scenario where we can compare INRPP with not only singlehomed transport protocols, such as TCP and RCP, but also multihomed transport protocols, such as MPTCP transport protocol. The third scenario, is a more complex hierarchical scenario, where we can evaluate INRPP performance in a similar scenario that can be used for the service placement UMOBILE module (introduced in D3.3, Section 4.4), where communications take place in the Internet domain between central servers and service instances placed in the edge of the network.

Dumbbell topology with a detour path

We first evaluate a simple scenario using a dumbbell topology with a fully-connected core component (nodes 0, 1, and 2), depicted in Figure 7. The purpose is to show in detail the operation of the different INRPP mechanisms in a simple setup. This topology has a bottleneck link (link 0-2) with 10Mbps capacity, but it also has another 10Mbps capacity one-hop detour path (link 0-1-2) that can be used to extend the bandwidth available at the bottleneck. Hosts have access links with 40Mbps bandwidth capacity, and links 4-0 and 2-3 have more capacity than the rest of the links. We pair the senders (three hosts on the left) and the receivers (three hosts on the right), and each sender initiates a single flow to its receiver pair. The three flows from the senders are initiated with one second gap in between. Each flow has the same size of 10 MB, and the size of each router cache is set to only 1.25 MB in order to demonstrate the activation of the backpressure mechanism, which eventually propagates to the sender. S_n^{high} is set to 1 MB and S_n^{low} to 500 KB. We set the packet size to 1500 bytes. Router interfaces buffer size is set to 50 ms [17] worth of traffic using Drop Tail and link latencies are 5 ms. In case of INRPP the L_{in}^{high} is set 40 ms and the L_{in}^{low} is set to 20 ms. We simulated the scenario using INRPP, RCP and TCP. In this scenario, we do not evaluate MPTCP since no hosts are multihomed, and therefore there is no possibility of establishing multiple sub-flows between peers.

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

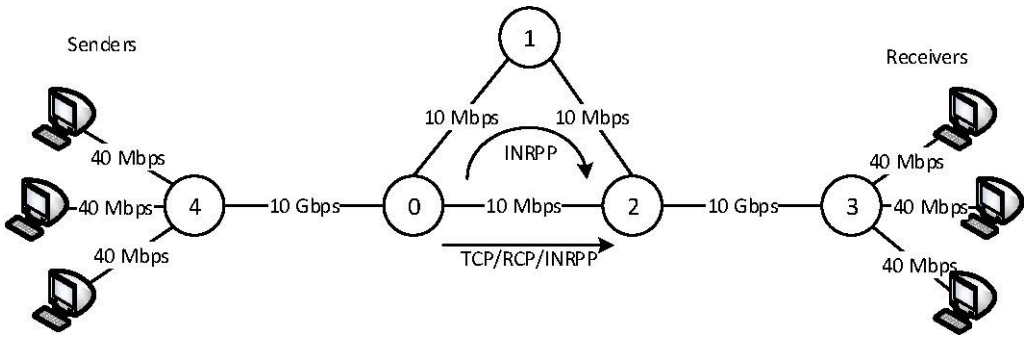


Figure 7 - Detour simple scenario

Protocol	AFCT	Fairness	Fairness different RTTs
INRPP	10.50s	0.9945	0.9972
RCP	25.96s	0.9934	0.9994
TCP	19.08s	0.8902	0.8321

Table 5- Dumbbell topology simulation results

In Table 5, we can observe the average flow completion time (AFCT) [18]. We see that INRPP can complete the flows much faster than TCP and RCP, because INRPP is using all the bandwidth available in the bottleneck, plus the bandwidth available in the detour path, when TCP and RCP is only using the bottleneck link capacity. In Figure 8, we demonstrate the goodput at the receiver in bps. With INRPP (top figure), we observe that the bandwidth is shared equally between the existing flows and there is no fluctuation when a new flow arrives. Particularly, the first flow starts at second 1 and is transmitted at 20Mbps using both the bottleneck link and the detour path shown with an arrow in Figure 7. When the second flow starts at second 2, the capacity is immediately shared between the two active flows (10Mbps each), while when the third flow starts at 3 seconds the available bandwidth is immediately split between the three active flows (≈ 6.66 Mbps per flow). When flows start completing, the existing flows adapt their rate and share the bandwidth no longer used by the completed flow. With TCP (bottom figure), on the other hand, we can observe that the goodput at the receiver is erratic and fluctuates excessively. TCP shares the bandwidth equally; however, it needs time to adapt to the new flow arrivals. Even after all the flows begin, the goodput oscillates continuously throughout the simulation due to the saw-tooth behaviour of the congestion window of TCP. In contrast, RCP does not have such oscillation in goodput, but we see that RCP also requires time to adapt and share the bandwidth between flows efficiently. In this simple scenario, RCP's slow adaptation to arriving flows is more pronounced due to the small

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

number of flows: RCP shares the bandwidth among flows by estimating the number of active flows, and when the number of active flows is small, this estimation is less accurate; that is, the error rate is higher when the estimation is, for instance, off by one. The slow adaptation of RCP to arriving flows leads to worse AFCT than TCP in this scenario.

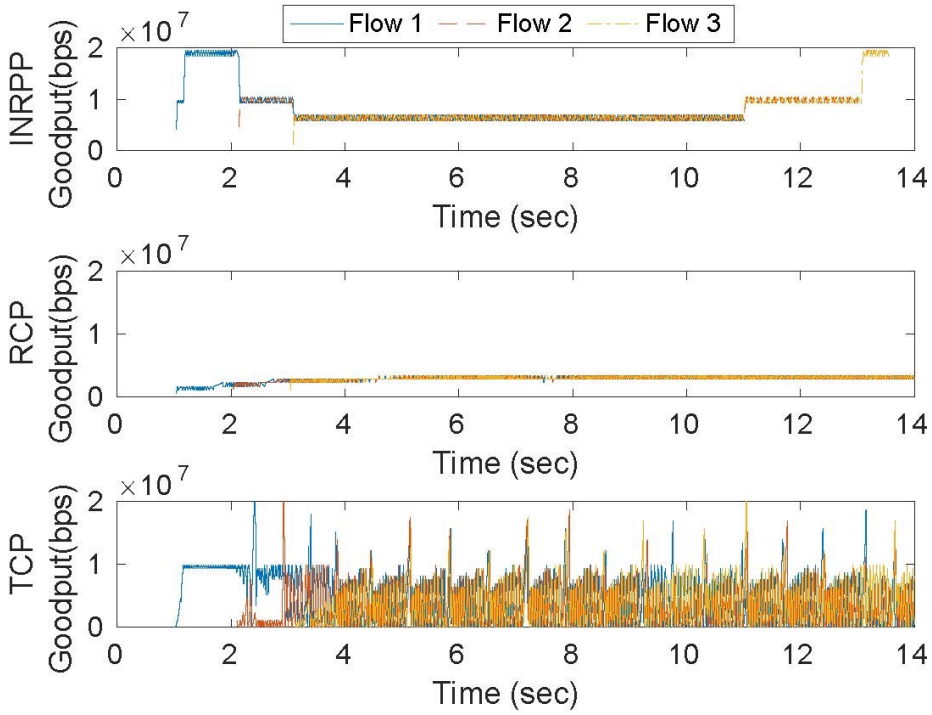


Figure 8 - Receiver goodput

In Table 5, we can also observe the fairness during the simulation. To measure the fairness, we first compute the Jain's index [19] on the instantaneous throughput of flows, sampled every 10 ms throughout the simulation, and then take the average of the samples to obtain a single average fairness value, shown in the third column. In order to demonstrate that INRPP is not affected by diverse RTT paths [20], we also evaluated fairness for heterogeneous access link latencies: 50, 100 and 200 ms. With homogeneous RTTs, we observe that INRPP and RCP fairness is close to the optimal with TCP having the worst fairness because of throughput oscillations. With heterogeneous RTTs, on the other hand, we observe that TCP fairness becomes even worse, while INRPP and RCP fairness is maintained close to 1, given their rate-based transmission pattern. In Figure 9, we can observe the cache occupancy, data input and output rates of nodes 0 and 4. The output data rate at node 0 is constant and equal to the capacity of the bottleneck link (0-2) plus the detour path capacity (2×10^7bps), when the interface of node 0 facing node 2 switches to *Store & Detour (S&D)* state from the initial *Push (P)* state, which happens shortly after the first flow starts. The input rate of node 4 initially decreases gradually when the senders of the first and the second flow complete their transmission, marked in the bottom plot with “Flow 1 transmitted" and “Flow 2 transmitted", respectively. After this point, only the sender of flow 3 is still transmitting, and around time 5s, the cache

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

occupancy exceeds the upper-bound, which causes node 4 to send a slow-down notification to the sender of flow 3. This causes the sender of flow 3 to enter closed-loop (CL) mode of operation, and therefore, slow-down around time 5s as shown in the bottom plot of the figure with the marker "CL flow 3". During CL, the incoming rate of node 4 reduces to the rate at which the packets of flow 3 leave the cache of the bottleneck node 2, i.e., 20Mbps/3 ≈ 6.66 Mbps, since the cache in node 0 contains data from all sources and shares its bandwidth equally among the flows. Later when the occupancy of the cache at node 4 drops below the lower-bound, it cancels the slow-down notification at the sender so it can send in *open-loop (OL)* mode again. However, at this point flow 3 has no more data to send, so we do not observe an increase in the rate of flow 3.

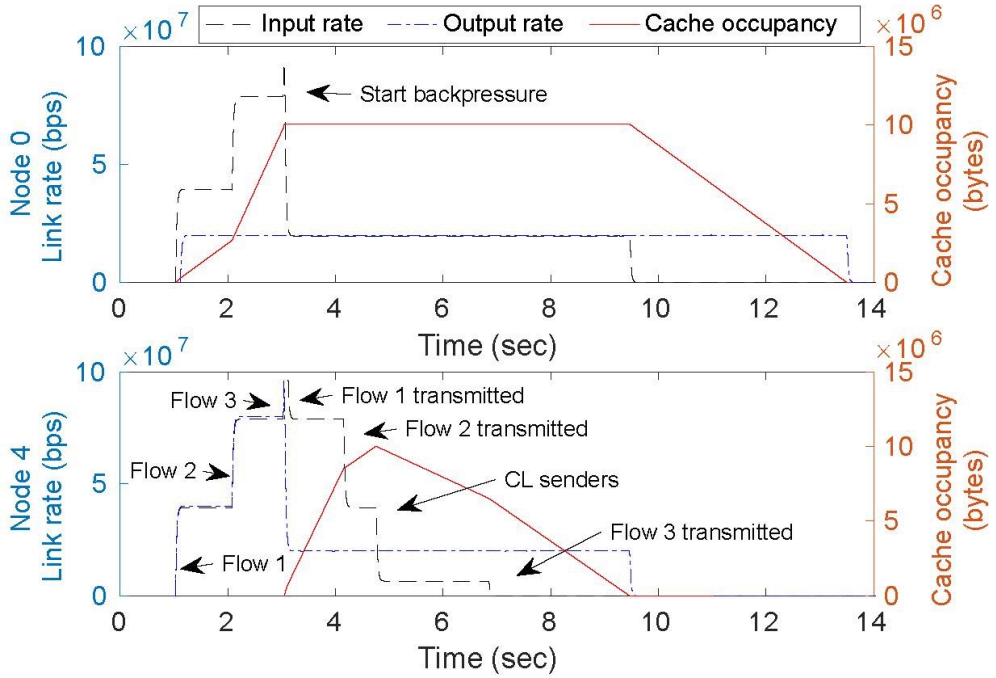


Figure 9 - Cache occupancy and data input/output rates (node 0 - top, node 4 - bottom)

In Figure 10 we depict the sequence number of the segments flowing through different locations of the network for a single flow. The first plot of the figure is the sequence number of the segments leaving the sender, where we observe how differently a flow is transmitted from an INRPP sender compared to TCP or RCP senders. In particular, in only 2 seconds the INRPP sender transmits the entire flow data of 10 MB at the rate of 40 Mbps. TCP and RCP senders are inserting data into the network in a closed-loop manner and are therefore transmitting for the whole duration of the flow, irrespectively of the available resources in the network. The second plot in the figure is the sequence number of the segments that are arriving at the cache of node 4. Here we can see that only the second half of the flow is cached in node 4; node 4 starts caching packets when it receives a slow-down request from node 0, which happens later (at 3.04s) after transmitting approximately half of the flow's data without caching. The third plot is the sequence number of the segments that are cached in node 0. In the beginning, the flow is cached at a higher rate, i.e., slope is steeper in the third plot until time 3.04s, because node 0 is

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

caching packets, but it is not in Backpressure state, and therefore the previous node (node 4) is transmitting at full rate. The bottom plot shows the sequence of segments arriving at the receiver host. At time 3.04s, node 0 gets into Backpressure state and sends a slow-down message to node 4. Node 0 implicitly performs ACK pacing as it sends ACKs at the rate of the bottleneck link (10Mbps). At this point node 4 gets into closed-loop mode, starts caching incoming packets and sends one data packet for every ACK received. Node 4's output rate, i.e., the input rate of node 0 shown in the third plot, quickly becomes equal to the rate at which the receiver host receives the flow. This can be observed by the similarity of the slopes in plot 3 after time 3.04 with the slope of plot 4 after that time.

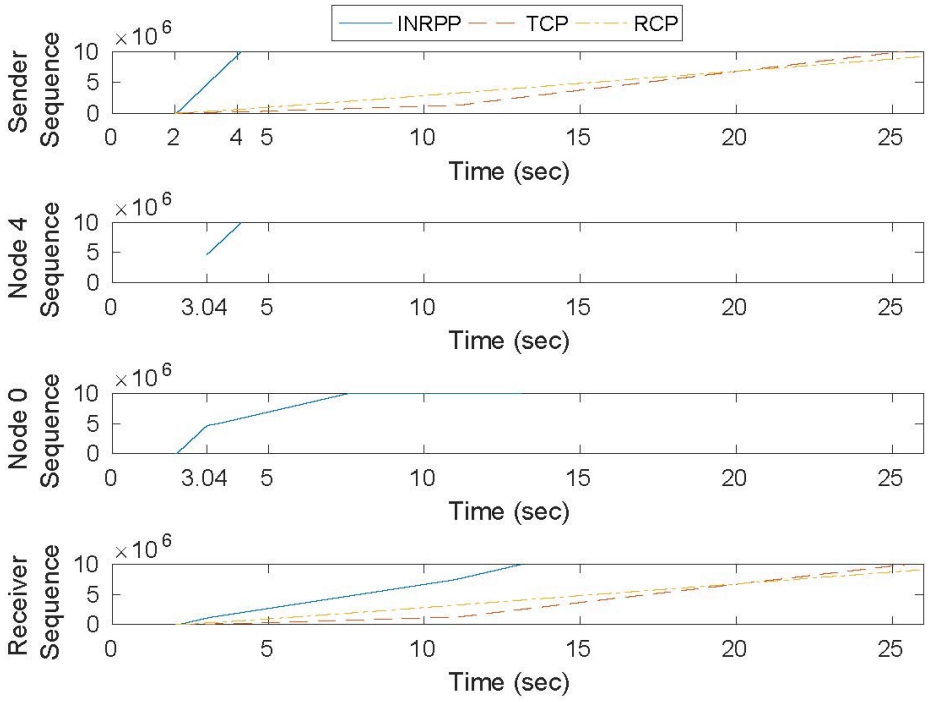


Figure 10 - Flows sequence number transmission

Figure 11 shows the Total Cache occupancy, as well as the contribution of each flow to the cache at node 0 (top) and node 4 (bottom). In this figure, we can observe that the allocation rate at the cache by each flow (sections of the flow plots with positive slope) and the flushing rate of the data belonging to each flow (sections of the flow plots with negative slope) is the same. The latter demonstrates that INRPP is fair because all flows are transmitted at the same rate. The top plot shows the state of node 0's interface (facing node 2) along with the cache occupancy. When the interface is backpressuring the previous node (node 4), the cache size is maintained in the upper-bound since the caching and the flushing rate is the same. The gradient in the slope of the cache occupancy of a flow increases as other flows complete. For example, the cache occupancy by flow 3 increases when flow 2 completes at time ≈ 8s in the top plot. The second plot in the figure shows the cache occupancy at node 4 after various events. Caching at node 4 starts at 3.04s when the node receives the slow-down message from node 0 and gets into closed-

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

loop operation. At time ≈ 4s, the arrival rate of flow 2 becomes 0 and the same happens later around time ≈ 5s for the flow 3.

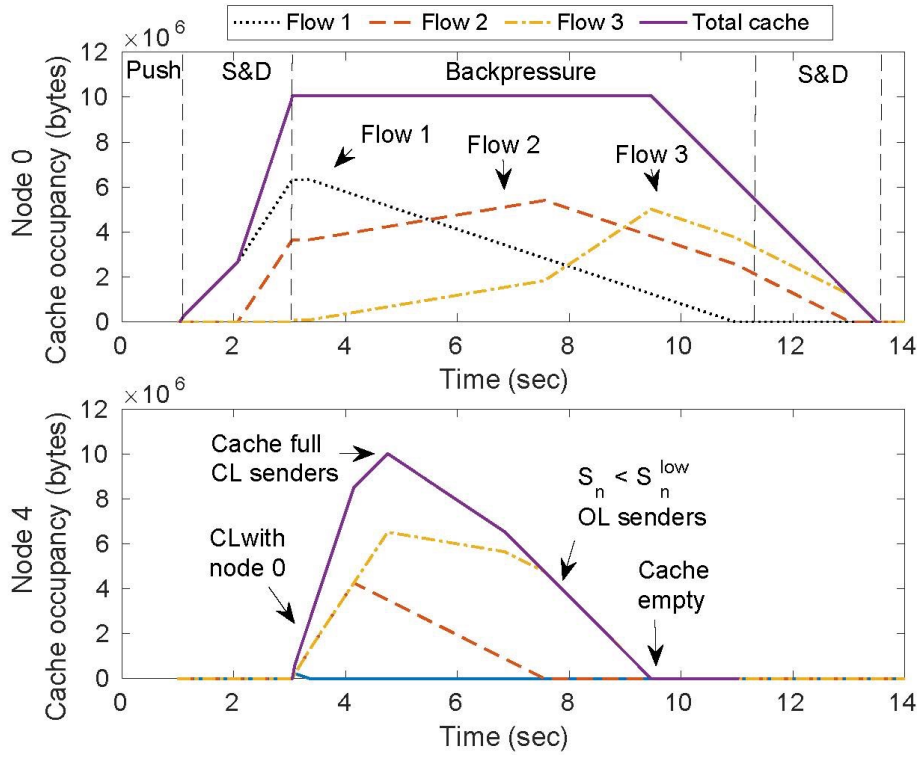


Figure 11 - INRPP cache occupancy

In Figure 12, we present the CDF of the packet interarrival times at the receiver application to examine the possible impact of packet reordering. We observe that INRPP interarrival time is constant between 1 and 4 ms, while some TCP packets are transmitted to the receiver with a delay up to ≈ 70 ms caused by drops and timeouts. This means that reordering in INRPP does not have severe impact as packets arrive without huge interarrival time gaps. Since there are no retransmissions due to losses in INRPP, the only cause of reordering is detouring of packets. However, even with detouring INRPP interarrival time is bounded and constant. This is because INRPP utilises a detour path only when there is no congestion on the detour path.

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

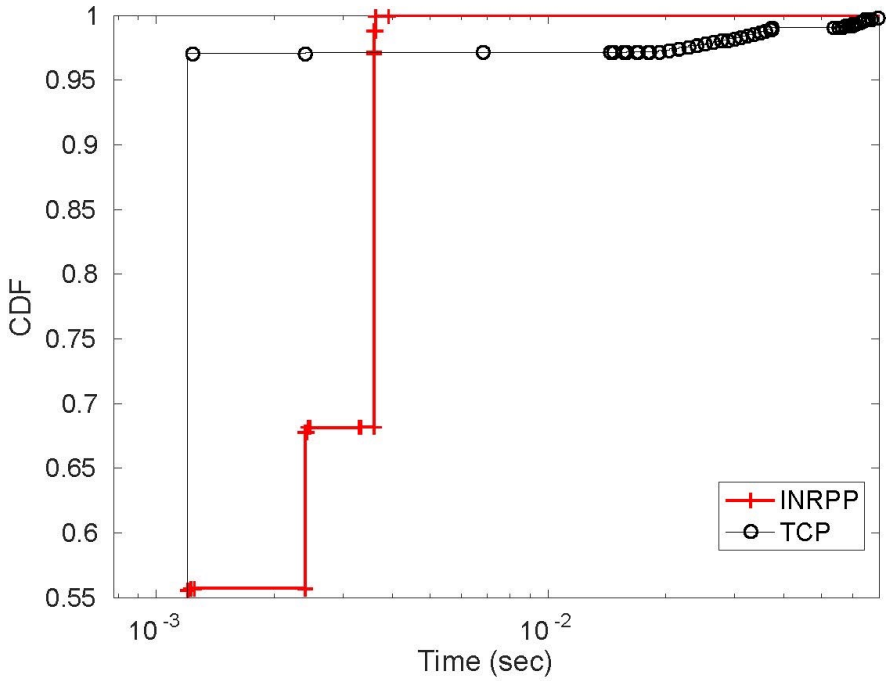


Figure 12 – CDF of packet interarrival times at the receiver

Multihomed topology

In this second scenario, we aim at evaluating INRPP in a multihomed scenario where we can compare it with a multipath transport protocols, such as MPTCP. MPTCP can exploit more than one path and establish multiple sub-flows to take advantage of available sub-paths. It is, however, constrained to end-host multihoming and can therefore, take advantage of e2e paths only. We evaluate the scenario depicted in Figure 13 where two paths can be used in parallel when using MPTCP (link 0-1 and link 2-1). MPTCP senders are multihomed and are connected to node 0 and 2 at the same time, while the MPTCP receivers are singlehomed and connected to node 1. MPTCP users establish two sub-flows, one for each pair of source, destination IP addresses. In addition, we also evaluate INRPP/TCP/RCP connecting the senders to node 0 and the receivers to node 1 only. This way, TCP and RCP will use only the path across link 0-1 (the shortest path) and INRPP will use this path as the main option, but will also be able to use the detour paths 0-3-1 and 0-2-1. The network parameters are the same as in Section 4.1, however, here, we increase the cache size (S_n^{high}) to 12.5MB (size proportional to the link bandwidth equivalent to 10 seconds of traffic); the end-points have 100 Mbps links.

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

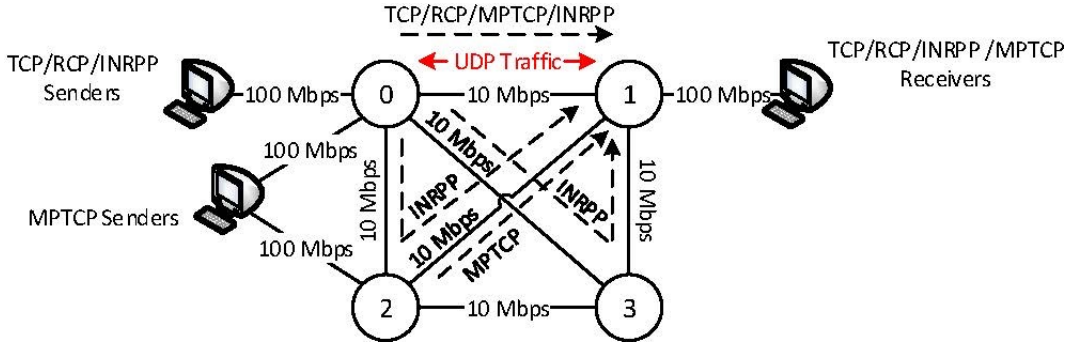
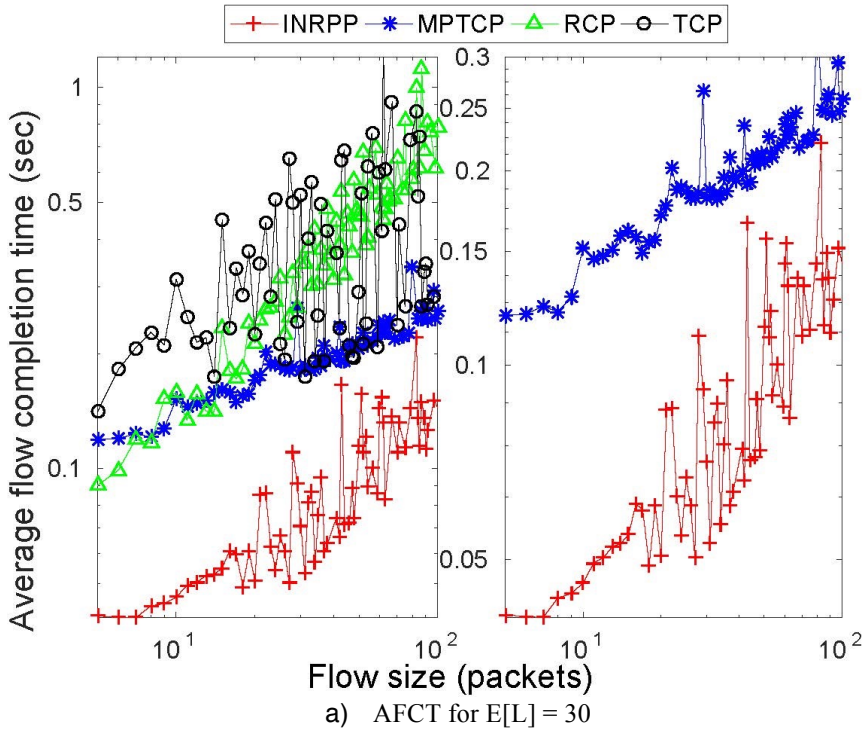


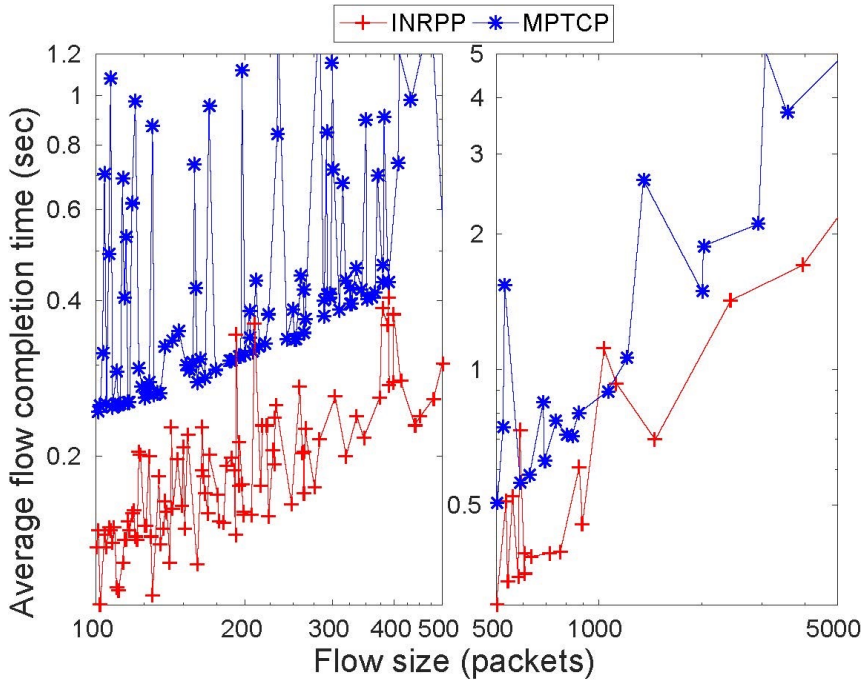
Figure 13 - Multihomed topology scenario

We evaluated this scenario using a Poisson Pareto Burst Process (PBPP) [21] to model Internet traffic. We used 1000 flows with poisson arrivals with a λ rate determined by the offered load of the network ρ where $\rho = \lambda \times E[L]/C$ ($E[L]$ is the average flow size and C the capacity of the link), that we set to 0.9. Flow sizes are pareto distributed with shape equal to 1.2. Note that this scenario is beneficial for MPTCP because it is a symmetric scenario. MPTCP behaviour can be worse when using highly asymmetric paths (in terms of bandwidth or latency) (such as 3G and WiFi) because of issues caused by disordering [22, 23]. In some of the simulations we also added cross-traffic in the link 0-1. Cross-traffic is added through UDP flows that come in every 15 secs, consume half of the bandwidth (5Mbps) for 5 secs and then leave.



a) AFCT for E[L] = 30

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



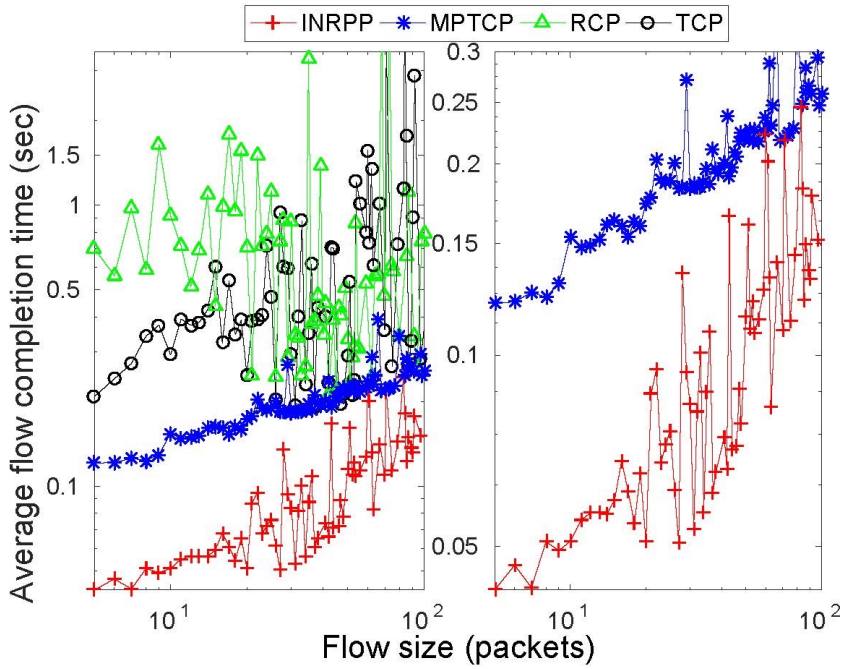
b) AFCT for E[L] = 500

Figure 14 - Multihomed AFCT w/o cross traffic

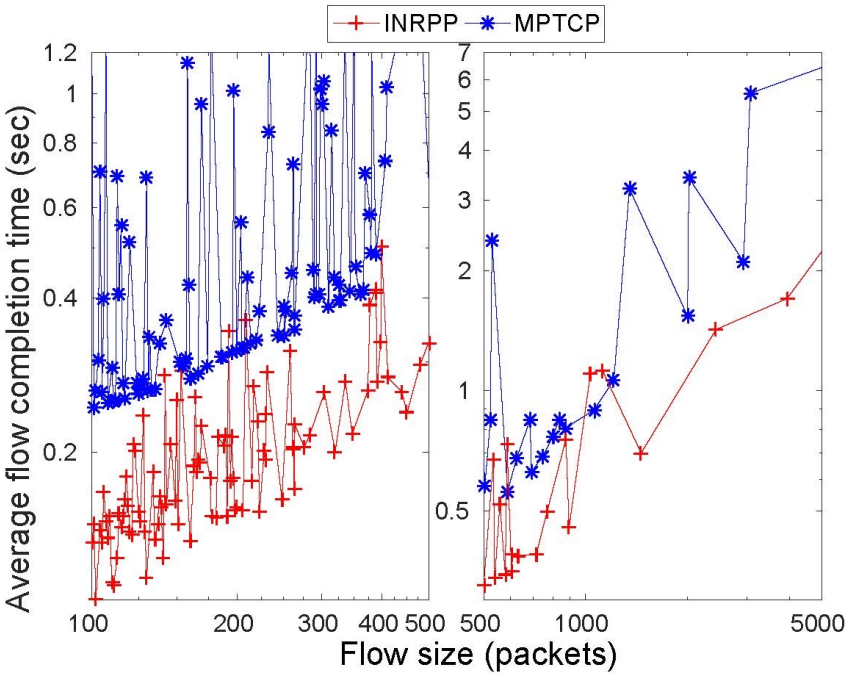
Fig. 14 shows the AFCT for different flow sizes when there is no cross-traffic in any of the links. We present the AFCT for all protocols using $E[L] = 30$, and the same results showing only MPTCP and INRPP at different scale. In Fig. 14(b), we present results using $E[L] = 500$ but only comparing MPTCP and INRPP. In Fig. 14(a), we observe that the AFCT of both MPTCP and INRPP are much lower than TCP or RCP. In this case, TCP does not differ significantly from RCP since we have a larger number of flows in the simulation and RCP's estimation of number of active flows is more accurate. However, for long flows RCP does not outperform TCP because the number of active flows is still not large enough for RCP to reduce its error rate in estimating the number of active flows.

The performance of RCP and TCP is substantially inferior compared to MPTCP and INRPP, because neither RCP nor TCP can use more than one path to send data. When comparing only MPTCP and INRPP we can see that INRPP clearly outperforms MPTCP, providing shorter flow completion times (up to around 50% in some cases), and therefore using the network resources more efficiently than MPTCP. First of all, MPTCP is not able to use all the detour paths available. MPTCP can use more resources than TCP, but at the same time it also inherits its limitations: First of all, MPTCP is an end-to-end resource pooling mechanism, and therefore, cannot exploit mid-path resources as INRPP does with the residual bandwidth available in detour paths. Secondly, AIMD-based MPTCP faces drops and timeouts, that most of the time does not imply a significant increase in AFCTs, but mainly causes poorer fairness performance - see Table 6. In fact, the chances of packet drop and timeouts in MPTCP increases linearly with the number of sub-flows. This is shown by the substantially worse fairness performance in case of short flows (see $E[L] = 30$ in Table 6).

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



a) AFCT for E[L] = 30



b) AFCT for E[L] = 500

Figure 15 - Multihomed AFCT with cross-traffic in link 0-1

Fig. 15 shows the AFCT, but this time with the cross-traffic pattern previously explained. In Fig. 15(a), we present the AFCT for all protocols using $E[L] = 30$ and in Fig. 15(b), we provide results using $E[L] = 500$ for MPTCP and INRPP only. In this figure, we observe again that INRPP significantly outperforms all other protocols. In this case, the flow completion time of RCP and TCP clearly gets worse due to the cross- traffic. However, the AFCT increment because of the cross-traffic, using MPTCP or INRPP, is almost

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

imperceptible compared with RCP and TCP. When using RCP or TCP, the cross-traffic generates more load than the link can absorb. However, when using MPTCP or INRPP, alternative path(s) (which do not use link 0-1) can be used to forward traffic in excess. Therefore, the multipath capability diminishes the impact of cross-traffic on AFCT.

In Table 6, we show the average fairness for the previous simulation. In all cases, INRPP presents the best performance, except for $E[L] = 500$, where RCP provides only marginally better fairness than INRPP.

Protocol	No cross-traffic		With cross traffic	
	$E[L] = 30$	$E[L] = 500$	$E[L] = 30$	$E[L] = 500$
TCP	0.8205	0.8575	0.4321	0.8883
RCP	0.9301	0.9589	0.7569	0.9959
MPTCP	0.6103	0.8947	0.6104	0.8515
INRPP	0.9298	0.9895	0.8225	0.9897

Table 6 – Average fairness using random flow sized in the multihomed topology

Hierarchical topology

In this last scenario, we evaluate INRPP in a network with transit-stub hierarchy, where different sets of users are clustered in the edges, and edges connect each other through a highly connected core. Different edge nodes (E) in Fig. 16, are connected to a transit network consisting of core (C) routers. Some of the edge nodes in the topology are interconnected to other edge nodes to form a small stub network. All the links have the same capacity of 10 Mbps except the links connecting the edge nodes of the servers to the core nodes, which have 100 Mbps capacity. In this scenario, we can have multiple bottlenecks and multiple detour paths in the network. More importantly, this scenario presents a more challenging environment for INRPP because the traffic flowing on shortest paths occupy all the links in the topology as opposed to previous two scenarios where the detour paths were exclusively used by detour traffic. Here, we only compare INRPP against RCP, TCP, and not MPTCP since there are no multihomed users.

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

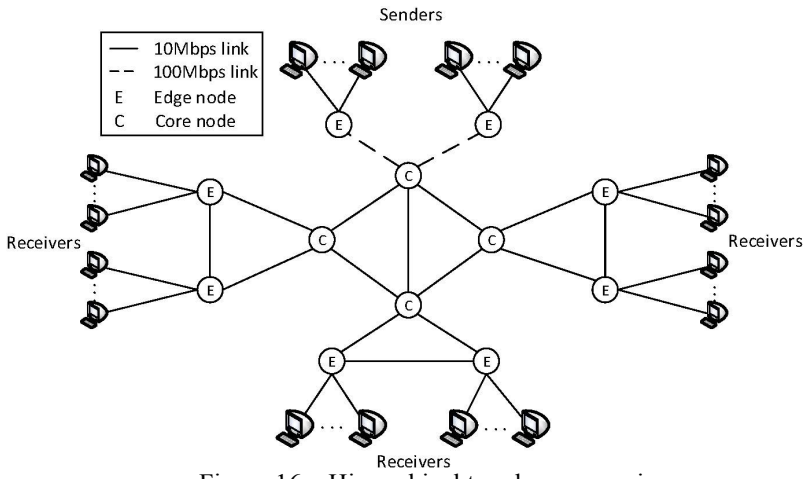


Figure 16 – Hierarchical topology scenario

Each group of senders (top of the figure) connected to a single edge node consists of 3000 hosts, and each group of receivers (left, right and bottom of the figure) connected to a single edge node consists of 1000 hosts. We randomly pair each sender with a receiver and start a flow from each sender to its pair, which makes a total of 6000 flows. In this scenario, we start these 6000 flows with an offered load of the access link $\rho = 0.8$ and $E[L] = 100$.

In Figure 17, we show the AFCT for varying flow sizes, using the same PBPP process, described in Section 4.2. We observe that even in this challenging scenario, INRPP clearly outperforms RCP (by at least 50%) and TCP (by more than 100%) in terms of flow completion time. It does so because INRPP can take advantage of available residual bandwidth even when it is available for very short time intervals, e.g., milliseconds, to detour excess traffic. We can conclude that dealing with congestion locally using INRPP is better than the e2e congestion control used by RCP and TCP given that the topology has detour paths (which is the case - see Table 1) and nodes possess caches.

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

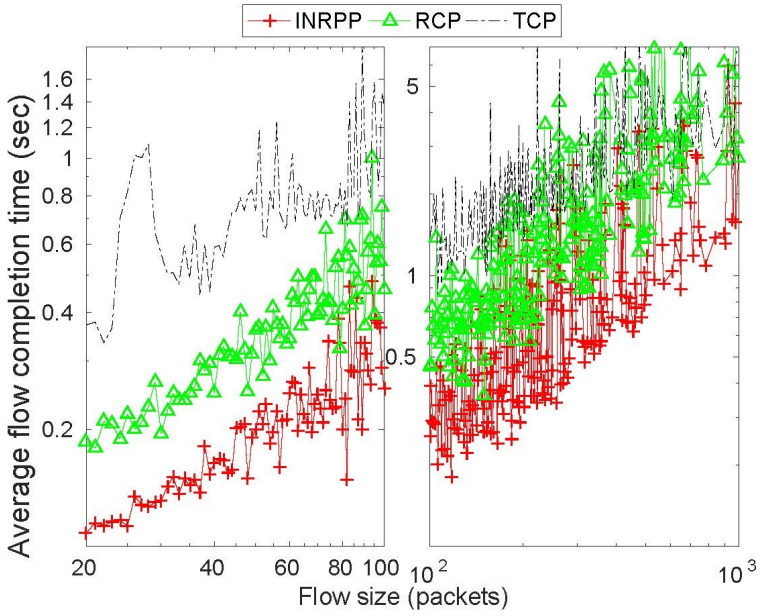


Figure 17 – AFCT for E[L] = 100 in the hierarchical topology scenario

4. INRPP over UMOBILE NDN networks

UMOBILE wireless Domain is expected to provide support for several scenarios with different network conditions, such as urban, remote or disaster areas, where users have specific applications and (maybe) infrastructure to allow communication among them, and such communication may follow a host-based (i.e., towards a specific host) or content-based (i.e., towards different interested parties) approach. These heterogeneous network conditions do not encourage an end-to-end transport solution, since it is very difficult to find a stable end-to-end path between two peers or the content provider and the requester.

This way, we plan to build a hop-by-hop congestion control, based on INRRP using NDN communications to be used in the UMOBILE wireless Domain. In this section, we propose next steps to implement INRPP over NDN-based wireless networks and we identify possible issues, that will be completed in D4.2 with a full implementation.

The central idea is to take advantage of the built-in abstractions (transparencies) provided by the ICN paradigm which are fully supported by the NDN implementation. Examples of these abstractions are named-based routing, in-network caching, decoupling of content from IP addresses, public/subscribe communication that decouples senders from receivers, multicast communications, data mobility and data replication.

4.1 INRPP NDN design principles

As a first step to implement INRPP over NDN, we assume Interest packets and Data or Content chunks generated by receivers and senders, respectively. Data receivers will request data at the application rate, and for bulk data transfers an initial processor sharing rate will have to be set. After receiving the first few chunks of data, the receiver will continuously adjust its requesting rate to the incoming data rate. Applications will

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

request for the immediate next chunk of data plus some anticipated data that the application is going to request in the near future.

The format of the Request Packets will be: $< N_c, ACK_c, A_c >$, where N_c is the next chunk requested by the application, ACK_c is an acknowledgment for the latest chunk received and A_c is the number of the last anticipated chunk. A_c, similarly to the initial request rate, is also a constant parameter set globally.

Data senders should keep state for each flow (similar to TCP senders) and operate in one of two modes. In the push-data mode, the objective of senders is to send as much data as their outgoing links can carry according to the N_c and A_c values of the requests. In the back-pressure mode, data senders slow-down their sending rate and enter a closed-loop mode of operation, where they send data at the rate with which they receive requests (1-to-1 flow balance).

Intermediate nodes should have two main functionalities routing/forwarding and caching. Nodes should forward data to their outgoing interfaces according to the interface's speed, therefore links always remain fully utilised. Each interface of the router can keep track of the requests that it has forwarded upstream (towards the source) for all other interfaces, per unit time. That is, each interface calculates the following ratio for each of the rest of the interfaces of the router:

$$y_{i\rightarrow i} = \frac{Reqs\ for\ I_i}{Reqs\ for\ \sum I_{i-}}$$

where I_i is interface i and I_{i-} are the rest of the interfaces. According to this value each interface can know the amount of traffic that it will receive for each of the rest of the interfaces in the next time interval T_i. A central management entity would need to collect all values from all interfaces and calculates the amounts of traffic that each of the interfaces will have to forward in the next T_i. We call this value Anticipated Rate for Interface i, or $r_i^{(\alpha)}$ and the actual rate with which interface i can forward traffic (i.e., the link capacity/speed) is denoted by r_i. In order to anticipate congestion, in NDN we can use $r_i^{(\alpha)}$ to know the expected traffic in an interface. Each interface can be in one of the INRPP states, depending on the $r_i^{(\alpha)}$. If $r_i^{(\alpha)} < r_i$, demand does not exceed supply and therefore, the link can deal with the expected amount of traffic, the node is in *Push* state. When $r_i^{(\alpha)} \approx r_i$ or $r_i^{(\alpha)} > r_i$, then demand is expected to exceed supply (within T_i) and the node will switch to *Store and Detour* state, and afterwards to the *Backpressure* state, if the detour phase finds that there is no alternative path to forward the data in excess, and finally needs to transmit to the upstream node the slow-down message.

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

4.2 INRPP implementation identified issues

However, implementing INRPP congestion control in an NDN wireless environment, such as the UMOBILE domain, presents multiple issues. In the following we briefly describe some issues that we identified to implement INRPP over NDN for the UMOBILE wireless and we will need to further investigate in the final version of the flowlet congestion control deliverable (D4.2):

- *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. As mentioned in the INRPP NDN design principle, the receiver continuously adjusts its requesting rate for sending Interest packets to fetch the incoming data. Upon detection of a congested link between an intermediate node and the receiver, the intermediate node sends a notification to the receiver to instruct it to reduce its requesting rate. The intermediate node can apply the Interest notification [32] pull model by appending information of the flow state into the interest name (i.e., the data is embedded in the string format). The result is that the receiver is able to proactively adjust the requested interest rate to avoid the congestion. 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.
- *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).
- *DTN opportunistic links*: As described in D3.1, UMOBILE provides an integrated information-centric delay-tolerant architecture. In particular, UMOBILE supports a forwarding method for opportunistic communications based on DTN tunneling (using the IBR-DTN implementation) which is exploited when the selection of the DTN face seems to be the best choice (e.g. in terms of cost). On the one hand, DTN tunnelling hinders 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

mechanism, since detouring and backpressure assumes undisruptive communications. On the other hand, the selection of the DTN forwarding approach, can be exploited to shift content distribution in time (i.e. suspend the transmission of non-critical data until the network is no longer congested). We aim to further investigate whether detouring and backpressure mechanisms can still be used along with the DTN face, as well as how the delay-tolerant forwarding approach can be used as a congestion control mechanism.

- *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.
- *Pull multicast [33]*: 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.

5. Conclusions

In this document, we defined an initial approach of the UMOBILE flowlet congestion control compatible with TCP/IP networks. Our extensive performance evaluation shows that INRPP does not risk network stability since it gets in a closed-loop mode when network conditions deteriorate. That said, however, when the network is less congested, INRPP takes immediate advantage of all the available bandwidth on both the main path and any detour available along the path. End-host clients do not need major modifications, while routers need to be equipped with caches and implement the detour and backpressure mechanisms. We believe that, given the performance gains of INRPP, the required changes are not prohibitive. In this document we also included an initial investigation to adapt the INRPP congestion control to NDN networks compatible with the UMOBILE wireless Domain. We emphasise that the plan is to take 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

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

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.). Yet we are only at a preliminary stage, we will explore these issues and the NDN built-in transparencies further in subsequent deliverables.

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

References

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

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

[3] 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.

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

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

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

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

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

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] "NS-3, A Discrete Event Simulator." http://www.nsnam.org.

[16] "Kickass for NS-3 - v. 1.0." http://users.eecs.northwestern.edu/~mef294/projects/kickass/code/README-kickass-ns3.

[17] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," SIGCOMM Comput. Commun. Rev., vol. 34, pp. 281–292, Aug. 2004.

[18] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," SIGCOMM Comput. Commun. Rev., vol. 36, pp. 59–62, Jan. 2006.

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

[19] 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.

[20] E. Gavaletz and J. Kaur, "Decomposing rtt-unfairness in transport protocols," in Local and Metropolitan Area Networks (LANMAN), 2010 17th IEEE Workshop on, pp. 1–6, May 2010.

[21] M. Zukerman, T. D. Neame, and R. G. Addie, "Internet traffic modeling and future technology implications," in INFOCOM 2003. Twenty-Second

Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 1, pp. 587–596 vol.1, March 2003.

[22] J. Iyengar, P. Amer, and R. Stewart, "Receive buffer blocking in concurrent multipath transfer," in Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE, vol. 1, pp. 6 pp.–, 2005.

[23] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tuxen, "On the use of concurrent multipath transfer over asymmetric paths," in Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pp. 1–6, Dec 2010.

[24] 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.

[25] 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.

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

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

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

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

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

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

[32] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton, "Named data networking: a natural design for data collection in wireless sensor networks," in Wireless Days (WD),IFIP, 2013.

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

[33] P. K. Chrysanthis, V. Liberatore, and K. Pruhs. Middleware for scalable data dissemination. In Q. H. Mahmoud, editor, Middleware for Communications, chapter 10, pages 237–260. John Wiley & Sons, 2004.

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in USENIX NSDI, 2011.

[41] M. Zhang et al., "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in USENIX ATEC, 2004.

[42] C. Raiciu et al., "Improving datacenter performance and robustness with multipath TCP," in ACM SIGCOMM, 2011.

[43] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle," SIGCOMM Comput. Commun. Rev., vol. 38, pp. 47–52, Sept. 2008.

[44] M. Honda, E. Balandina, P. Sarolahti, and L. Eggert, "Designing a resource pooling transport protocol," in IEEE INFOCOM workshops, pp. 13–18, 2009.

[45] C. Hopps, "IETF RFC 2992, analysis of an equal-cost multi-path algorithm," 2000.

[46] J. He and J. Rexford, "Toward internet-wide multipath routing," Network, IEEE, vol. 22, no. 2, pp. 16–21, 2008.

[47] X. Liu and L. Xiao, "A survey of multihoming technology in stub networks: Current research and open issues," Netwrk. Mag. of Global Internetwkg., vol. 21, pp. 32–40, May 2007.

[48] C. Lumezanu, D. Levin, and N. Spring, "PeerWise Discovery and Negotiation of Faster Paths," in ACM HotNets-VI, 2007.

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

[49] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in ACM SIGCOMM, 2005.

[50] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-path vs. multi-path overlay routing," in ACM SIGCOMM IMC, pp. 91–100, 2003.

[51] R. Kokku, A. Bohra, S. Ganguly, and A. Venkataramani, "A multipath background network architecture," in IEEE INFOCOM, pp. 1352–1360, 2007.

[52] S.-J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu, "Bandwidth-aware routing in overlay networks," in IEEE INFOCOM, pp. 1732–1740, 2008.

[53] P. Key, L. Massouli´e, and D. Towsley, "Path selection and multipath congestion control," Commun. ACM, vol. 54, pp. 109–116, Jan. 2011.

[54] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y.Le Boudec, "Mptcp is not pareto-optimal: Performance issues and a possible solution," in Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '12, (New York, NY, USA), pp. 1–12, ACM, 2012.

[55] C. Raiciu, M. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols." RFC 6356 (Experimental), Oct. 2011.

[56] Y. Thomas, G. Xylomenos, C. Tsilopoulos and G. C. Polyzos, "Multi-flow congestion control with network assistance," *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, Vienna, 2016, pp. 440-448. doi: 10.1109/IFIPNetworking.2016.7497200

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