Revisiting Resource Pooling: The Case for In-Network Resource Sharing

Ioannis Psaras, Lorenzo Saino and George Pavlou
Dept. of Electrical & Electronic Engineering
University College London
WC1E 7JE, Torrington Place, London, UK
{i.psaras, l.saino, g.pavlou}@ucl.ac.uk

ABSTRACT

We question the widely adopted view of in-network caches acting as temporary storage for the most popular content in Information-Centric Networks (ICN). Instead, we propose that in-network storage is used as a place of *temporary custody for incoming content in a store and forward manner*. Given this functionality of in-network storage, senders push content into the network in an open-loop manner to take advantage of underutilised links. When content hits the bottleneck link it gets re-routed through alternative uncongested paths. If alternative paths do not exist, incoming content is temporarily stored in in-network caches, while the system enters a closed-loop, back-pressure mode of operation to avoid congestive collapse.

Our proposal follows in spirit the *resource pooling principle*, which, however, is restricted to *end-to-end* resources and paths. We extend this principle to also take advantage of in-network resources, in terms of multiplicity of available sub-paths (as compared to multihomed users only) and in-network cache space. We call the proposed principle *In-Network Resource Pooling Principle* (INRPP). Using the INRPP, congestion, or increased contention over a link, is dealt with locally in a hop-by-hop manner, instead of end-to-end. INRPP utilises resources throughout the network more efficiently and opens up new directions for research in the multipath routing and congestion control areas.

Categories and Subject Descriptors

C2.1 [**Computer-Communication Networks**]: Network Architecture and Design- Distributed Networks

Hotnets-XIII, October 27–28, 2014, Los Angeles, CA, USA.

General Terms

Design, Algorithms, Performance

Keywords

ICN; Congestion Control; In-network Caching

1. INTRODUCTION

Information-Centric Networking (ICN) dismantles the (e2e) host-to-host communication model and proposes direct communication between the end-user and the *content or information* itself [2], [25].

A fundamental feature of ICNs is *native content naming* ([2], [10], [25], [30]), which in turn enables in-network caching [25]: since the network is not *content-agnostic*, but it is transferring explicitly named objects, data can be temporarily cached in routers (instead of being merely buffered) and be later re-sent to one or multiple requesting users. Furthermore, the receiver-oriented *request-response* model of ICN (contrary to the sender-driven *DATA-ACK* model of traditional TCP) gives the opportunity to *approximate the expected incoming link load on a given router interface per unit time* (*i.e.*, every request is asking for one chunk of data in the opposite direction) [55]. In-network caching has received considerable attention lately in the context of ICN, but solely in the direction of improving cache hit performance by optimising cache [9], [44] and content [40] placement, request-to-cache routing [16], [46] and cache router design [4], [51].

In this paper, we look at in-network router caching from a different angle and question whether caches can take on alternative roles and assist with the control of network congestion and in-network resource management. We propose that *routers act as temporary custodians*[1] *for incoming content along a given path from source(s) to destination(s).* We work to integrate this design foundation into the recently proposed *resource*

[1]The term "custodians" is used in disconnected networks in a similar way to guarantee that nodes accepting data will make sure to deliver the data to its destination (or to a next hop towards the destination) [49].

pooling principle (RPP) [52], according to which *"a collection of networked resources behave as though they make up a single pooled resource. The general method of resource pooling is to build mechanisms for shifting load between various parts of the network"* [52]. Resource pooling, as proposed in [52] and further investigated in [22] and [53] has so far focused on shifting load between outgoing links of end-users only, making use of *Multipath TCP* [53]. RPP in its current form ([22], [53]) is restricted to *end-to-end bandwidth resources* (see Fig. 2(ii)). For this reason we refer to the above definition as *e2eRPP* and investigate the possibility of complementing *e2eRPP* with an *In-Network Resource Pooling Principle (INRPP)* (see Fig. 1), largely defined as a group of mechanisms based on which *load is shifted along the entire delivery path in a hop-by-hop manner, utilising all available subpaths and in-network caches* (see Fig. 2(iii)).

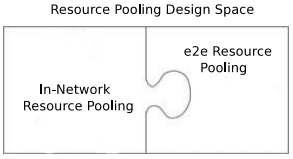


Figure 1: Network Resource Pooling

In-Network Resource Pooling (INRP) is based on the concept of *dealing with increased demand for network resources locally and not only in an end-to-end manner*. The INRP paradigm comprises three different phases: *i)* **push-data:** content is pushed as far in the path as possible in an open-loop, processor sharing manner [14], based on the path's hop-by-hop bandwidth resources and not on acknowledged data from the end-user; pushed data consists of both requested and anticipated data (data not explicitly requested yet) to take advantage of underutilised links; *ii)* **detour:** when pushed data reaches the bottleneck link, it is split in *flowlets* [50], which are then detoured through alternative paths towards the destination [20], [27]; *iii)* **back-pressure:** if alternative paths do not exist, data gets cached at the bottleneck router and the system enters a closed-loop, back-pressure mode of operation [48] to avoid extensive caching and congestion [39].

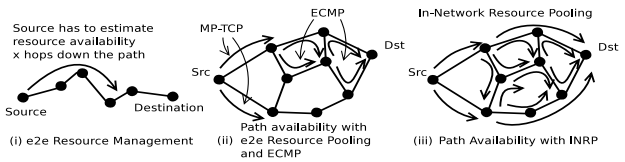


Figure 2: Single-/Multi-Path Resource Utilisation

This alternative view of in-network caching contradicts the closed feedback loop of TCP, as well as the resource probing nature of TCP's AIMD [24]. We question this common design principle of closed-loop control and investigate whether alternative, ICN-based concepts, can move traffic faster without causing packet drops.

We base our design on generic conventions for ICN operations, such as the *request-response* transmission cycle, but do not restrict the applicability of INRP to any of the existing ICN architectures (*e.g.*, [8], [12], [25], [54], or assumptions therein, *e.g.*, symmetric forward-return paths). We believe that with the required modifications INRP could apply to most ICN approaches, either native or future, evolutionary, IP-based ones. The design considerations discussed here should be considered as foundations for efficient resource management, which require further research in order to be made operational.

The distinct concepts that form the In-Network Resource Pooling Principle have been investigated in the past in different disciplines and with different objectives. Rate-based congestion control has been proposed in [34], [43] and has been shown to achieve promising results. Furthermore, detour mechanisms, multipath routing and traffic engineering are well-investigated topics [20], [21], [27]. Finally, there exists a fair amount of work on back-pressure techniques to better control sending rates [39], [48]. We discuss each of those items next, identify reasons why their applicability to date is limited and finally, attempt to put those different pieces together in an *In-Network Resource Pooling framework*.

2. BACKGROUND AND MOTIVATION

2.1 Congestion: A Network-layer Problem

Congestion is generally defined as the event upon which link utilisation approaches its maximum capacity, resulting in packets being dropped from overflowing output queues [19]. That said, congestion is by definition a *network-layer* phenomenon (*i.e.*, packets are being dropped), but has traditionally been dealt with at the *transport-layer*, in a closed-loop, *end-to-end* fashion (*i.e.*, only the end points can regulate the traffic inserted in the network). In a road network, this is similar to controlling the departure times of travellers depending on their routes, rather than the actual traffic itself along the route using traffic lights. The goal of a transport protocol is to fully utilise path resources from a sender to a receiver in a fair manner among all pairs of end-hosts sharing (part of) the same path [11].

The end-point driven control of network congestion, however, inevitably relies on probing algorithms to estimate the available resources down the delivery path and respond to congestion events in a reactive way (*i.e.*, after packets have been dropped). The ultimate goal of AIMD (and TCP) is to estimate the capacity of the lowest-throughput link along the delivery path and transmit traffic according to that. In essence, *TCP*

senders have to speculate on the conditions of the end-to-end path (Fig. 2(i)).

Although probing in the form of AIMD is shown to achieve fairness and stability, it comes with several drawbacks, which have concerned the research community for decades. These include reactive response to congestion, burstiness, RTT-unfairness, slow flow completion time, flow-synchronisation and inherent ability to clog the network with data and cause congestion [15], [43].

As a result, several techniques have been proposed to tackle these issues, from smoother transmission patterns [5], to rate-based (*e.g.*, RCP [14]) and hop-by-hop rate controllers [34], but also network layer, Active Queue Management (AQM) approaches [1]. All of those alternatives targeted *explicit and/or faster feedback to the sender regarding network conditions in order to overcome the slow, end-to-end feedback loop* [42].

Take-away I: *The end-point-based closed-loop congestion control of TCP is bound to inefficient performance due to decisions made at the two ends of the TCP connection (i.e., far from the congested area).*

2.2 Multipath Congestion Control

Several studies have investigated the benefits of utilising multiple paths to send traffic from a source to a destination [28]. Multipath TCP (MPTCP) [53] has received wide attention recently, since by making use of its *resource pooling principle*, it can exploit available resources along multiple paths.

MPTCP, similarly to any other multipath congestion control approach (*e.g.*, mTCP [56]), requires that sources of data are multihomed, a straightforward requirement to exploit multiple paths *in an end-to-end fashion* [32]. Given that currently multihoming is not common among residential users, MPTCP seems to be finding more space for deployment in data-center environments [41]. Recent trends, however, reveal that multihomed stub domains increase rapidly, and that with IPv6, multihoming becomes even easier [36]. This fact is interesting for one more reason: 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. In a multihomed, or FTTH environment, however, the bottleneck will inevitably move to the core of the network causing more severe congestion events.

Indeed, the ever increasing rate of TCP's AIMD is set to clog the network and its buffers no matter how fast the link is.

Take-away II: *Multipath congestion control based on TCP/AIMD gives the opportunity to exploit more available resources, according to the resource-pooling principle. Multipath congestion control is still, however, restricted to utilise e2e paths only, hence exhibits the same inefficiencies as normal TCP.*

2.3 Multipath Routing

Pushing the bottleneck at the edge of the network, as discussed in the previous section, results in relatively stable aggregate rates in the core, despite the bursty, saw-tooth behaviour of TCP/AIMD. Multipath routing has therefore been used *in the core* (as opposed to multipath congestion control, which applies to the edge only) for traffic engineering and load-balancing reasons [27], [33]. Several techniques have been proposed for forwarding on multiple paths both at the intra- and the inter-domain level [20], [32]. Flexible forwarding can be applied either per packet or per flow (see ECMP [23], flowlet cache [20], or flowlet switching [50]).

Detour techniques on the other hand have been proposed for overlay networks [29], mainly at the AS level [21] to bypass the IP/BGP routing instructions which do not take into account latency or bandwidth characteristics of paths [3]. In all cases, however, and although network routers have a better view of both the network conditions and the (intra-domain) topology [20], it is still left to the end-points to regulate the sending rates or choose which of the available paths to follow [35].

Take-away III: *Despite extensive studies on multipath routing and multipath congestion control, 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.*

3. IN-NETWORK RESOURCE POOLING

3.1 Framework Overview

Fundamentally, there exists a very tight relationship between the end-to-end, closed-loop feedback system of TCP and the size of the system's buffers [13]: the end-to-end, closed-loop feedback necessitates small buffers along the delivery path. This relationship is in turn bound to *move traffic along a path as fast as the path's slowest link*. Given the single-path nature of TCP, moving traffic according to the path's slowest link guarantees *global stability* (*i.e.*, stability along the *e2e* path through *e2e* rate-adaptation). Fairness on the other hand, is guaranteed *locally* (*i.e.*, based on the capacity of the bottleneck link).

We argue against this relationship and in the spirit of INRPP propose that: *i) stability should be local*, and *ii) fairness should be global*. *Local stability* demands that the node before the bottleneck link takes appropriate action when conditions deteriorate. *Global fairness* on the other hand requires that all resources (both bandwidth and cache) up until the bottleneck link are shared equally among participating flows.

Consider two flows in the topology of Fig. 3. 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 $F = \frac{\sum(T)^2}{n\sum(T^2)}$, where T is each flow's throughput and n is the total number of flows, the system fairness in this case is 0.73. In case more than one flows traverse the bottleneck link (2-4), they would share equally the available bandwidth (local fairness).

In contrast, according to the In-Network Resource Pooling Principle introduced here (right part of Fig. 3), the shared link (1-2) is split equally among the two flows (global fairness). Node 2 has two options in this case: *i)* find alternative routes to reach node 4 (local stability), or *ii)* enter backpressure mode and notify node 1 to reduce its sending rate (closed-loop system to avoid extensive caching at node 2). In the topology of Fig. 3, node 3 can accommodate the extra 3Mbps. In this case, Jain's index indicates perfect system fairness.

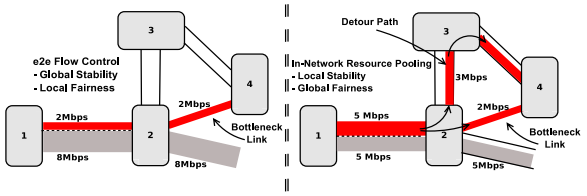


Figure 3: Left: *e2e* Flow Control: Bandwidth is split according to the slowest link on the path. Right: *INRPP:* Bandwidth is split equally up to the bottleneck link (global fairness). Detour applies to guarantee local stability.

The current system would fail to achieve the above goals, as: *i)* caches cannot be operational in a *content-agnostic* network, where data containers (*i.e.*, IP packets) are buffered, forwarded based on the destination IP address and then discarded (as they are of no use any longer), and *ii)* end-points have to approximate the available resources along the whole path (according to a closed-loop, ever-increasing, probing manner).[2]

3.2 End-Point Node Model

Under an ICN environment, we assume *Request Packets* and *Data or Content Chunks* generated by receivers and senders, respectively.

Data Receivers request data at the application rate. For bulk data transfers (*i.e.*, ftp), an initial rate will have to be set, similar to the initial window of TCP. After receiving the first few chunks of data, the receiver continuously adjusts its requesting rate to the incoming data rate.

[2]A workaround for *local stability and global fairness* could be for every node in the network to run a separate instance of TCP for every flow it is serving and keep per flow queues (similarly to Split-TCP). This, however, would be rather expensive in terms of processing delays.

Applications 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 Packet* is: $\langle N_c, ACK_c, A_c \rangle$, 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.

For our discussion here, we do not consider lost chunks, or out of order delivery of chunks and rather focus on how to engineer global fairness and local stability. Techniques to overcome out-of-order packets in TCP have been proposed in [22] and [50] and could also be used in the context of INRP. Note, however, that out-of-order delivery in our case is not a sign of congestion. Instead lost packets are identified by explicit timers or NACKS.

Data Senders 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 this phase, senders multiplex flows in a simple processor sharing fashion [14]. 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).

3.3 Network Router Functionality

Intermediate routers have two main functionalities, namely, *routing/forwarding and caching*. Routers do not maintain per-flow queues, but have a scheduler which multiplexes data from incoming to outgoing interfaces in a round-robin fashion. Routers forward data to their outgoing interfaces according to the interface's speed, therefore links always remain fully utilised.[3]

Each interface of the router keeps 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:

$$y_{i^- \rightarrow i} = \frac{Reqs\ for\ I_i}{Reqs\ for\ \sum I_{i^-}} \quad (1)$$

where I_i is interface i and I_{i^-} are the rest of the interfaces. According to this value each interface knows the amount of traffic that it will receive for each of the rest of the interfaces in the next time interval T_i.[4] A

[3]The forwarding speed can be set to a value smaller than the outgoing interface's maximum speed to avoid operating at full capacity and be able to accommodate bursts.

[4]Setting the time interval T_i is not trivial (similarly to T_d in TeXCP [27]), as this value will have to take into account diverse RTT paths. To avoid keeping per flow state at each router, a reasonable setting for T_i would be the average RTT of data chunks. This can be sampled from the timestamp that data chunks carry on them.

central management entity of the router is collecting all values from all interfaces and calculates (sums up) 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 $r_i^{(a)}$ and the actual rate with which interface i can forward traffic (*i.e.*, the link capacity/speed) is denoted by r_i. Each interface can be in one of the following three phases:

Push-data Phase: If $r_i^{(a)} < r_i$, demand does not exceed supply and therefore, the link can deal with the expected amount of traffic.

Detour Phase: When $r_i^{(a)} \rightsquigarrow r_i$, or $r_i^{(a)} > r_i$, then demand is expected to exceed supply (within T_i) and alternative techniques have to be applied to avoid extensive congestion. We note that demand will exceed supply because of increased demand from end-users towards this part of the network and not because of transport protocol's increasing rates, *i.e.*, senders are forwarding data according to the available capacity at their outgoing interfaces [14].

In the *detour phase* the router will have to seek alternative paths to forward data towards its destination. Note, however, that the router estimates the expected traffic at the request phase, and therefore, has a time window of approximately $T_i/2$ (*i.e.*, $T_i \approx avgRTT$, depending on the distance ratio of the router between source and destination) to assign traffic to paths before the actual traffic arrives.

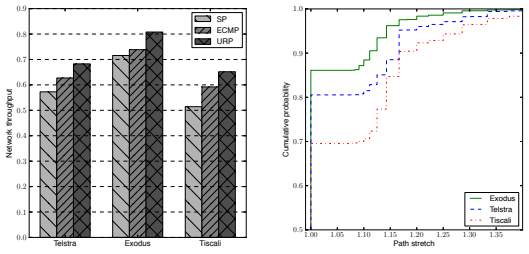
Our initial approach here suggests intra-domain detouring at the router level. Indeed, we find that real network topologies can on average provide one-hop detour paths on more than 50% of links (reaching up to 92% for Level-3), two-hop detour paths on 30% of links, and three-plus hop detour paths on less than 5% of links (see Table 1). In our sample of nine topologies detour is not available for approximately 13% of links only. To detour through a specific path, the router would have to spoof the destination router's identifier with that of the node back on the original path (effectively tunnelling through the detour node [29]).

Table 1: Available Detour Paths in Real Topologies

ISP	1 hop	2 hops	3+ hops	N/A
Exodus (US)	49.77%	35.48%	6.68%	8.06%
VSNL (IN)	25.00%	33.33%	0.00%	41.67%
Level 3	92.22%	6.55%	0.68%	0.55%
Sprint (US)	56.66%	37.08%	1.81%	4.45%
AT&T (US)	34.84%	61.69%	0.72%	2.74%
EBONE (EU)	50.66%	36.22%	6.30%	6.82%
Telstra (AUS)	70.05%	10.42%	1.06%	18.47%
Tiscali (EU)	24.50%	39.85%	10.15%	25.50%
Verio (US)	71.50%	17.09%	1.74%	9.68%
Average	52.80%	30.86%	3.24%	13.10%

Upon detouring and without any extra information on the load of the links on the detour path, data may find itself before another congested link. We plan to investigate two approaches to deal with this issue: *i)* nodes periodically communicate their average link utilisation between their one-hop neighbours (see similar techniques in [27], [29], [31]), or *ii)* nodes on the detour path also have the option to further detour if they see that the immediate path to the destination is also congested. The first approach suggests that routers keep state *per outgoing interface of their one-hop neighbours*. This is needed in order to forward towards this detour direction exactly as much traffic as this detour path can accommodate. In the example of Fig. 3, node 2 would know the available free capacity of the interface 3-4. Although this inevitably introduces extra overhead it would help make informed decisions in the detour phase.

We have evaluated the performance of the *push-data* and *detour* mechanisms of INRP in a simple flow-level simulator, where flows arrive Poisson distributed. Routers exploit up to 1-hop detours and nodes on the detour path can further detour, but for one extra hop only. If senders see extra available bandwidth they insert more data in the network. We compare the performance of the above abstraction of INRP against single-, shortest-path routing (SP) and Equal-Cost Multipath (ECMP) in three real topologies. Fig. 4a shows that INRP achieves between 9-15% extra bandwidth utilisation, compared to SP. We expect this to translate to faster flow completion time by the same proportion. ECMP also performs better than SP, as we do not consider bottlenecks at the edges of the network, which would result in stable aggregates at the core of the network. In Fig. 4b we observe that this is achieved with minimal path stretch and expect that in case of 2+ hop detours performance will improve further. The combination of these two results indirectly reveals that although a large number of flows swap between paths, they do not swap simultaneously to cause concurrent over-utilisation of some paths and under-utilisation of others.



(a) Overall throughput (b) INRP Path Stretch

Figure 4: *INRP* performance

Back-pressure Phase: Finally, if the detour phase finds that there is no alternative path to forward the data in excess, either because no link exists, or because detour links are congested as well, the interface gets into the *backpressure phase*. In this phase, the congested

node: *i)* caches incoming data, and *ii)* explicitly informs its one-hop upstream neighbour to forward data at a slower requested rate. Effectively, the system enters a closed feedback loop in order to avoid excessive caching at the congested node.

The rate of caching of incoming data depends on the incoming link's speed (see [4] and [40] for representative figures). Values reported there indicate that a 10GB cache after a 40Gbps link can hold incoming traffic for 2 seconds - much more than the average RTT (and timeout) in the Internet today. Note that caching here does not replace buffering (incurring huge queuing delays), as during the push-data and detour phases, the senders forward more data than the client has requested. Effectively, intermediate nodes temporarily store future/anticipated data that the user has not requested yet.

In turn, the upstream neighbour node that has been informed of the congested link has two options: *i)* find more-than-one-hop detour to bypass the congested node, or *ii)* forward the explicit backpressure notification further back the path to reach the data sender. In the first case, the node gets into the detour mode, as if one of its own interfaces was congested.[5] In the second case, where the notification is sent all the way back to the sender, the latter is entering a closed-feedback loop for this flow. This further implies that the sender will have to re-divide the available bandwidth between the rest of the flows (again in a processor sharing manner) on its outgoing interface to utilise all available resources.

4. SUMMARY AND CONCLUSIONS

Existing proposals for an ICN Internet require substantial modifications to the current architecture, both from a conceptual perspective and from an infrastructure one (*e.g.*, extra layers in CCN/NDN [25], topology managers in PURSUIT [54], DHT-based name resolution in NetInf [12], mediation planes in COMET [8], [38]). We believe that although these changes require investment both in capital and in research terms, the exercise is worth taking, given the benefits of an ICN-based Internet. We also believe that if the exercise is to be undertaken, the full potential of the shift should be considered. The feasibility of the In-Network Resource Pooling Principle provides a good first step on this direction.

[5]Again here, keeping state of the outgoing interfaces of the one-hop neighbours would help nodes exploit bandwidth for the non-congested interfaces of their neighbor nodes. For example, in Fig. 3, if node 2 sends a back-pressure message to node 1 (*e.g.*, the detour through node 3 is not available), then node 1 would either slow down the traffic for both flows (if it doesn't know which outgoing interface of node 2 is congested), or would slow down traffic for the flow that goes through the bottleneck only (if it keeps state per outgoing interface of node 2).

Recent efforts in the ICN transport-layer area have mainly focused on adjusting the main mechanisms of TCP and AIMD to fit to an ICN environment (*e.g.*, [6], [7], [26], [37], [45], [47], [55]). Closer to our work are [26], [45] and [55]. Although the protocol in [55] 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 discussed above. Furthermore, detouring in [55] takes place at the *Interest* phase (instead of the data phase), hence, its accuracy is bound to be outdated by approximately $RTT/2$. The work in [45] on the other hand, is based on hop-by-hop rates to shape the rate of interests and therefore, data as well (similarly to [34]). This would inevitably require per-flow queues to regulate the rate of each flow according to the path it is traversing and transmit traffic based on the path's slowest link (in a *global stability* fashion). Instead, INRP would require (at most) state per outgoing interface of neighbour nodes. 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.

The open issues of INRPP outnumber the resolved ones, but our initial investigations show that it indeed achieves the objectives of *local stability* and *global fairness*. Deployment issues [17] and co-existence with TCP/IP will have to be investigated [18]; out-of-order delivery will need to be dealt with [50]; and monitoring mechanisms at the interface level will need to be finalised to enable stable detouring and avoid extensive link swapping [27]. These are some of the immediate future directions to assess the full potential of *INRPP*.

5. ACKNOWLEDGMENTS

This work was supported by UK EPSRC COMIT project, grant no. EP/K019589/1 and by the EU-JAPAN initiative EU FP7/NICT GreenICN project, grant no. (EU)608518/(NICT)167. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the EC, or NICT.

6. REFERENCES

[1] R. Adams. Active queue management: A survey. *IEEE Communications Surveys Tutorials, 2012*.
[2] B. Ahlgren et al. Design considerations for a network of information. In *ACM ReArch*, 2008.
[3] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan. Best-path vs. multi-path overlay routing. In *ACM SIGCOMM IMC*, pages 91–100, 2003.
[4] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *ACM ReArch*, 2010.
[5] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *IEEE INFOCOM*, 2001.
[6] 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.

[7] 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.

[8] W. K. Chai and et al. CURLING: Content-ubiquitous resolution and delivery infrastructure for next-generation services. *IEEE Communications Magazine*, 49(3):112–120, 2011.

[9] W. K. Chai, D. He, I. Psaras, and G. Pavlou. Cache less for more in information-centric networks (extended version). *Elsevier Computer Communications Journal, Special Issue on Information-Centric Networking*, 36(7):758–770, 2012.

[10] D. Cheriton and M. Gritter. TRIAD: A New Next-Generation Internet Architecture, 2000.

[11] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1 – 14, 1989.

[12] C. Dannewitz et al. Network of information (NetInf): An information-centric networking architecture. *Elsevier COMCOM*, 36(7):721 – 735, 2013.

[13] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer sizing for congested internet links. In *IEEE INFOCOM*, 2005.

[14] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor sharing flows in the internet. In *IWQoS'05*, pages 271–285.

[15] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM CCR*, 36(1):59–62, Jan. 2006.

[16] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga. CATT: potential based routing with content caching for ICN. In *ACM SIGCOMM ICN*, pages 49–54. ACM, 2012.

[17] B. Ford and J. Iyengar. Breaking up the transport logjam. In *ACM HotNets-VII*, 2008.

[18] B. Ford and J. Iyengar. Efficient cross-layer negotiation. In *ACM HotNets-VIII*, 2009.

[19] J. Gettys and K. Nichols. Bufferbloat: dark buffers in the internet. *Commun. ACM*, 55(1):57–65, Jan. 2012.

[20] J. He and J. Rexford. Toward internet-wide multipath routing. *Network, IEEE*, 22(2):16–21, 2008.

[21] S. W. Ho et al. Deconstructing internet paths: an approach for AS-level detour route discovery. In *IPTPS'09*.

[22] M. Honda, E. Balandina, P. Sarolahti, and L. Eggert. Designing a resource pooling transport protocol. In *IEEE INFOCOM workshops*, pages 13–18, 2009.

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

[24] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM*, 1988.

[25] V. Jacobson et al. Networking Named Content. In *ACM CoNEXT '09*, pages 1–12, 2009.

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

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

[28] P. Key, L. Massoulié, and D. Towsley. Path selection and multipath congestion control. *Commun. ACM*, 54(1):109–116, Jan. 2011.

[29] R. Kokku, A. Bohra, S. Ganguly, and A. Venkataramani. A multipath background network architecture. In *IEEE INFOCOM*, pages 1352–1360, 2007.

[30] T. Koponen and et al. A Data-Oriented (and beyond) Network Architecture. *ACM SIGCOMM*, 2007.

[31] S.-J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu. Bandwidth-aware routing in overlay networks. In *IEEE INFOCOM*, pages 1732–1740, 2008.

[32] X. Liu and L. Xiao. A survey of multihoming technology in stub networks: Current research and open issues. *Netwrk. Mag. of Global Internetwkg.*, 21(3):32–40, May 2007.

[33] C. Lumezanu, D. Levin, and N. Spring. PeerWise Discovery and Negotiation of Faster Paths. In *ACM HotNets-VI*, 2007.

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

[35] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. In *ACM SIGCOMM*, pages 27–38, 2008.

[36] E. Nordmark and M. Bagnulo. Shim6: Level 3 multihoming shim protocol for IPv6, 2009.

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

[38] G. Pavlou, N. Wang, W. Chai, and I. Psaras. Internet-scale content mediation in information-centric networks. *Annals of Telecommunications*, 2012.

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

[40] I. Psaras, W. K. Chai, and G. Pavlou. In-network cache management and resource allocation for information-centric networks. *IEEE Transactions on Parallel and Distributed Systems (TPDS), in press*, 2014.

[41] C. Raiciu et al. Improving datacenter performance and robustness with multipath TCP. In *ACM SIGCOMM*, 2011.

[42] K. K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *ACM Trans. Comput. Syst.*, 8(2):158–181, May 1990.

[43] D. Ros and M. Welzl. Less-than-best-effort service: A survey of end-to-end approaches. *Communications Surveys Tutorials, IEEE*, 15(2):898–908, 2013.

[44] D. Rossi and G. Rossini. On sizing CCN content stores by exploiting topological information. In *IEEE INFOCOM NOMEN*, 2012.

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

[46] L. Saino, I. Psaras, and G. Pavlou. Hash-routing schemes for information centric networking. In *ACM SIGCOMM ICN'13*, Hong Kong, China, Aug. 2013.

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

[48] S. Sarkar and L. Tassiulas. Back pressure based multicast scheduling for fair bandwidth allocation. In *IEEE INFOCOM*, volume 2, pages 1123–1132 vol.2, 2001.

[49] M. Seligman, K. Fall, and P. Mundur. Alternative custodians for congestion control in delay tolerant networks. In *ACM SIGCOMM CHANTS '06*.

[50] S. Sinha, S. Kandula, and D. Katabi. Harnessing TCP's burstiness with flowlet switching. In *ACM HotNets-III*, 2004.

[51] M. Varvello, D. Perino, and J. Esteban. Caesar: a content router for high speed forwarding. In *ACM SIGCOMM ICN*, 2012.

[52] D. Wischik, M. Handley, and M. B. Braun. The resource pooling principle. *SIGCOMM Comput. Commun. Rev.*, 38(5):47–52, Sept. 2008.

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

[54] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos. Caching and mobility support in a publish-subscribe internet architecture. *Communications Magazine, IEEE*, 50(7):52–58, 2012.

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

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