CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking

Lorenzo Saino, Cosmin Cocora and George Pavlou
Department of Electrical and Electronic Engineering
University College London
WC1E 7JE, Torrington Place, London, UK
Email: {l.saino, c.cocora, g.pavlou}@ee.ucl.ac.uk

***Abstract*—Content Centric Networking (CCN) is a recently proposed information-centric Internet architecture in which the main network abstraction is represented by location-agnostic content identifiers instead of node identifiers. In CCN each content object is divided into packet-size chunks. When a content object is transferred, routers on the path can cache single chunks which they can use to serve subsequent requests from other users.**

Since content chunks in CCN may be retrieved from a number of different nodes/caches, implicit-feedback transport protocols will not be able to work efficiently, because it is not possible to set an appropriate timeout value based on RTT estimations given that the data source may change frequently during a flow.

In order to address this problem, we propose in this paper a scalable, implicit-feedback congestion control protocol, capable of coping with RTT unpredictability using a novel *anticipated interests* mechanism to predict the location of chunks before they are actually served. Our evaluation shows that our protocol outperforms similar receiver-driven protocols, in particular when content chunks are scattered across network paths due to reduced cache sizes, long-tail content popularity distribution or the adoption of specific caching policies.

I. Introduction

While the Internet was originally designed for host-to-host communications, it has increasingly been used for content dissemination and retrieval, with video streaming alone accounting for well over half of the current Internet traffic [1]. This mismatch between the original design assumptions and current usage patterns has partially been addressed through application layer solutions such as Content Delivery Networks (CDN) and Peer-to-peer (P2P) overlays, which have retrofitted some desirable content-aware functionalities on top of the existing architecture. However, the lack of native network support for content distribution restricts the efficiency of such approaches, and also potentially hinders the evolution of the Internet as a whole. As a result, recent research efforts such as CCN [2], COMET [3], PSIRP/PURSUIT [4] and SAIL/4WARD/NetInf [5] have all focused on rethinking the Internet as an information-centric network.

Among these approaches, the architecture proposed by Content Centric Networking (CCN) has gained considerable momentum. In CCN content objects are assigned a location-agnostic identifier and partitioned into *chunks*, which can be contained within a single packet and cached at any router along a path. CCN operates using two complementary network primitives. Content objects are requested through Interest packets, which are routed according to the identifier of the requested content towards the closest available copy and content is then delivered through Data packets that follow the reverse route.

In content-oriented architectures where caching takes place at a chunk granularity, as in the case of CCN, it is possible that chunks may be delivered from different network nodes when ”streaming” an entire content object. This renders existing TCP-based congestion control mechanisms unusable in such scenarios for two reasons:

- Out-of-order delivery or variations in inter-arrival intervals can no longer be used as an indication of network congestion. In fact, a packet might arrive out of order simply because it has been sent by a node located further away than the source of the other packets.
- RTO estimation, as defined by Jacobson’s algorithm [6], does not span multiple data sources, and as such it cannot be used reliably.

One could speculate that the use of a single timeout value may not detriment network performance if chunks of the same flow are retrieved mostly from a single location or from caches located close to each other. However, the location from which chunks are retrieved depends strongly on a number of factors including caching policy, amount of existing cross-traffic, content popularity and cache sizes [7]. For example, heavy cross traffic, small cache sizes, large content population and adoption of caching policies such as age-based cooperative caching [8] and probabilistic caching [9] are all factors which contribute to scattering chunks across a path, exacerbating the inability of a single timeout value in tracking congestion.

Congestion control protocols designed specifically for CCN fail typically to address the fact that chunks may originate from different sources. ICP [10] and ICTP [11] for example calculate an Interest retransmission timer from previous RTT samples of Data packets, irrespective of origin. ConTug [12] maintains separate timeouts for each content source. However, it assumes that the receiver knows the location of each content chunk before the transfer starts and it does not change during the flow, which is not currently possible in CCN and it is not easily achievable without compromising scalability. Further proposals such as HoBHIS [13] and HR-ICP [14] require each CCN router to keep per-flow state information, which would strongly affect scalability and deployment in core Internet

routers.

In this paper we propose Content Centric TCP (CCTCP), a scalable alternative to existing proposals which takes into account content retrieval from multiple sources. CCTCP is an implicit-feedback receiver-driven transport protocol specifically designed for CCN, which operates by using *anticipated interests* to infer the location of chunks before they are requested, and keeps individual timeouts for each expected source.

Evaluation results demonstrate that protocol performance is independent of the caching dynamics. In addition, while it performs comparably to ICP when packets originate from nodes close to each other, it performs significantly better when content chunks are scattered across a path.

The remainder of this paper is organized as follows. Section II describes the protocol design of CCTCP. Section III provides the performance evaluation of CCTCP using ns-3 simulations. Finally, conclusions are drawn in section IV.

II. Protocol design

CCTCP is designed to address the following concerns:

- **Scalability**: any proposed solution should scale efficiently. For this reason, we designed CCTCP so that it does keep state information only at the receivers, with CCN routers remaining stateless, in accordance to the original Internet end-to-end principle
- **Incremental deployability**: the resulting protocol must be able to coexist with legacy traffic and in particular should be fair to existing TCP flows.
- **Independence from caching policies**: introducing dependencies between the performance of transport protocols and caching policies limits innovation in both. Therefore, in order to allow the evolution of the underlying CCN architecture, CCTCP has been designed with no assumption about the manner in which network caches insert and evict content chunks.

In order to ensure TCP fairness, CCTCP is derived from TCP New Reno [15] and similarly comprises *slow start* and *congestion avoidance* phases. However, in contrast to TCP, rate control is enforced by receivers instead of senders, through the regulation of Interest packets. Furthermore, a receiver may keep multiple congestion windows and timeout values for each flow: one for each CCN cache serving chunks in response to Interest packets. If all chunks originate from the same node, CCTCP operates as a receiver-driven implementation of TCP.

A. Anticipated interests and RTT estimation

In CCTCP, a Data packet not received within a timeout period is inferred as a sign of congestion. However, since a receiver maintains multiple timeout values for each flow, CCTCP must have a mechanism for estimating which node a Data packet will originate from in order to trigger the correct timeout. In our design, we propose that in each Interest packet issued, the requester lists the identifiers of a set of chunks that it intends to request closely after the current request is satisfied. These *anticipated interests* can be used by the receiver to predict the expected RTT of subsequent chunks with the support of CCN routers.

In order to achieve this, for every Interest packet received, a CCN router must verify if it is also storing any of the chunks the receiver intends to subsequently request. If the router has at least one of the future chunks but not the chunk to which the current request refers, it forwards the Interest packet to the next hop, appending to it:

- identifier(s) of the chunk(s) that are present in its cache,
- timestamp T_I when the Interest packet was processed,
- a unique node identifier, which is not necessarily routable

Upon forwarding an Interest packet, a router must not alter information on the availability of anticipated chunks appended by previous routers. Finally, if a router holds the requested chunk in its cache, it returns the corresponding Data packet to the requester, appending the anticipated chunks header retrieved from the Interest packet.

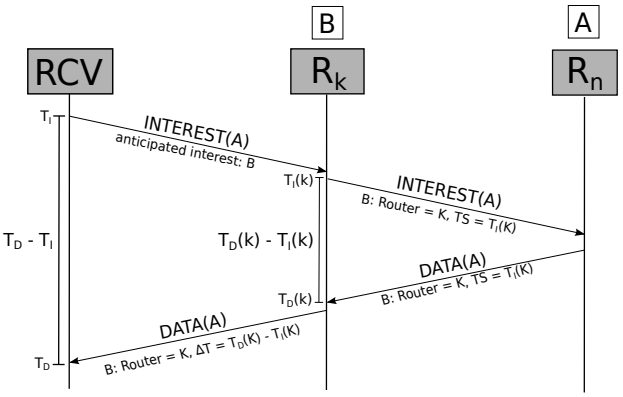


Fig. 1: RTT estimation and *anticipated interests* mechanism

In the return path of the Data packet, each router must analyse the content of the anticipated chunks header. If the header contains information the router appended to the Interest packet, then the router replaces the absolute timestamp T_I with the difference between the current time T_D, for when it processed the Data packet, and T_I. In addition, the router also appends to the Data packet a hop count initially set to 0 that will be incremented by other routers on the path to the receiver. This hop count can then be analyzed by the receiver to learn the topological order of caches on the path.

Using the information contained within a Data packet, a receiver can estimate the RTT between itself and router k which holds an anticipated chunk as follows:

$$RTT(k) = (T_D - T_I) - (T_D(k) - T_I(k)) \qquad (1)$$

where T_I and T_D are measured by the receiver itself respectively when it sends the Interest packet and when it receives the Data packet and $T_D(k) - T_I(k)$ is measured by router k and appended to the returned Data packet.

It should be mentioned that it is not necessary to have clock synchronization among network nodes in order to obtain accurate RTT measurements. Clock offsets do not change the result

of the equation above because any skew will be "cancelled out" in the subtraction operation. Moreover, this mechanism does not require routers to keep any state because information is self contained in the *anticipated interests* header.

Finally, it should be also noted that for this mechanism to work, Interest and Data packets must use the same path, which is the case in CCN and in most content-centric architectures. In addition, we assume that Interest packets are routed to the original source of the requested content and a cached copy serves this request only if it is stored in one of the routers located in the path between receiver and original source, which is exactly the way CCN works.

B. Window and timeout update algorithms

In addition to keeping as many timeout values as the number of expected sources of chunks, the CCTCP receiver also keeps multiple values of the congestion window (CWND) parameter, which evolves over time depending on whether Data packets are received before the timeout expiration or not.

To better explain how congestion windows and timeout values are updated during the protocol operations, we will use an example.

First, the receiver starts requesting the first chunk of the content object with an initial timeout value, arbitrarily set to 3 seconds for our simulations. When the related Data packet is received, it updates the timeout associated with the node returning the data as well as the RTT of the nodes "signalling" the presence of future chunks. The timeout update is executed according to Jacobson's algorithm [6] adopted in TCP New Reno. Similarly, the protocol also updates the CWND associated to the node which sent the packet, according to the TCP *slow start*. As Data packets are received, the receiver updates the values of timeout and CWND related to all nodes serving chunks of that content object. The CWND keeps increasing exponentially until a timeout expiration occurs. When this happens, the CWND is set to half of the current flight size, the timeout value is doubled and the CWND starts increasing linearly, according to TCP *congestion avoidance* phase.

It is important to highlight that when a Data packet is received, the receiver increases the size of the CWND associated with the node sending the successfully received Data packet as well as those associated with all nodes topologically located between the receiver itself and that node. The CWND of these intermediate nodes is increased as much as needed to make their data rate (i.e. $CWND/SRTT$) equal to the one associated to the sender of the Data packet. Similarly, when a Data packet is lost or received after the timeout expiration, the receiver halves the CWND and doubles the timeout value of all the nodes topologically located between the node from which the Data packet was expected and the original source of the content. The rationale behind this behavior is that if no congestion is experienced on the path between the receiver and a node k, there should be no congestion on any of its subpaths and, therefore, it is sensible to increase the congestion windows for all nodes in that path. Likewise, when congestion occurs on that path, then all its superpaths experience congestion as well and therefore it is reasonable to decrease the congestion window associated to all nodes beyond the one from which the lost packet was expected.

As a result of this behavior, at any time during the operation of the protocol, the data rate associated to node k on the path between the receiver and the original content source s is greater or equal than the data rate associated to any other node located further away than k. Precisely:

$$\frac{CWND(k)}{SRTT(k)} \geq \frac{CWND(k+x)}{SRTT(k+x)} \qquad \forall x \in [1, s-k] \quad (2)$$

Similarly, the timeout value RTO associated to a specific node is less than or equal to the timeout value associated to any node located further from the receiver.

$$RTO(k) \leq RTO(k+x) \qquad \forall x \in [1, s-k] \quad (3)$$

As a result of this joint update of all CWND values, at any time over the duration of a flow, congestion window values satisfy the following interesting properties (cfr. fig. 2):

- there is a set of adjacent caches contiguous to the receiver from which chunks are downloaded using the *slow start* window update algorithm.
- there is a set of adjacent routers contiguous to the original source from which chunks are downloaded using the *congestion avoidance* window update algorithm.

This occurs because of the following reason.

In the beginning of each flow, the congestion windows associated to all routers increase according to *slow start*. As timeout expirations start occurring, the nodes located further away than the cache which generated the loss drop their window and start entering *congestion avoidance*. As the flow continues, routers closer to the receivers will start experiencing losses and the boundary between the *congestion avoidance* and *slow start* regions will move towards the receiver.

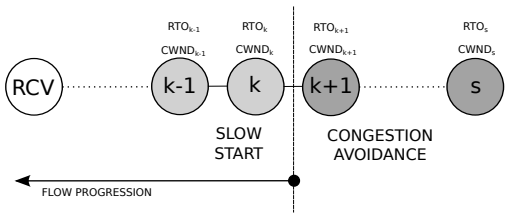


Fig. 2: *Slow start* and *congestion avoidance* regions

In conclusion, it is important to mention that receivers do not send Interest packets in a burst as in TCP New Reno, but in contrast, Interest packets are paced with the interval:

$$T_{send}(i) = \frac{SRTT(i)}{CWND(i)} \quad (4)$$

where i is the cache expected to serve the Interest request.

This mechanism has been extensively adopted by receiver-driven transport protocols in the past, such as WebTP [16]. The rationale behind this design choice lies in the fact that

Interest packets have a negligible size in comparison to Data packets and this is also the case with their transmission time. As a result, if Interest packets were sent in a burst to the same content source, Data packets associated to Interests located towards the end of the burst would experience greater RTT than those at the beginning. In fact, the delay experienced by the last packets is increased by the transmission time needed to insert previous Data packets on the communication channel. As a consequence, the RTT would have fluctuations that would compromise the efficiency of the RTO estimation. The pacing of Interest packets we have adopted addresses this problem and we validated its effectiveness in our experimentation.

C. Virtual Packet Loss

Another feature of CCTCP worth discussing is the *virtual packet loss* mechanism. As explained in section II-B, when a Data packet is lost or received after timeout expiration, the congestion windows associated to the expected source and all nodes further away in the path are reduced. However, this might not be sufficient to react to congestion fast enough. In fact, if the link where congestion occurs is very close to the receiver, it would be desirable to reduce also the congestion windows of the nodes located between the congestion point and the expected source, rather than just those of the expected source and all further nodes.

To mitigate this issue, we propose a mechanism that we called *virtual packet loss*, illustrated in fig. 3. Every time a Data packet is received, the receiver must compare the RTT associated to all intermediate nodes which signalled the availability of anticipated chunks with the timeout value associated to them, even if the Data packet has been received before the timeout for the expected source expired. If the estimated RTT associated to at least one of the intermediate nodes is greater than the associated timeout value, then the receiver treats this packet reception as if it was a packet loss. Therefore, it still keeps the chunk received, but decreases CWND and increases the timeout value for the node whose RTT exceeded the timeout value and all nodes beyond it.

This mechanism provides for early identification of congestion and therefore makes the protocol more efficient.

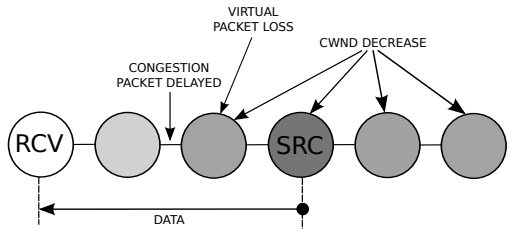


Fig. 3: Virtual packet loss mechanism

D. Special cases

1) No anticipated chunks information available: One problem that may occur in the protocol operation is that, despite the use of *anticipated interests*, the receiver may not have any information about the location of one or more chunks it is about to request. This may occur for many reasons, but most likely because none of the nodes topologically located between the receiver and the nodes previously responding to Interest packets holds a copy of the desired chunk.

In this situation, the protocol still operates correctly. In fact, it simply operates as if it were the first packet of a new flow, therefore with a CWND equal to 1. Naturally, this situation could affect the throughput if it occurs frequently. However, our experimentations showed that the occurrence of such event is rare and has a minimal impact on protocol performance.

2) Data packets originated from a different node than expected: Another problem that may occur is when the receiver knows where a chunk is located, but the content is served by another cache, located either closer or further than expected source. This event could occur for a number of different reasons, including cache eviction and cross traffic.

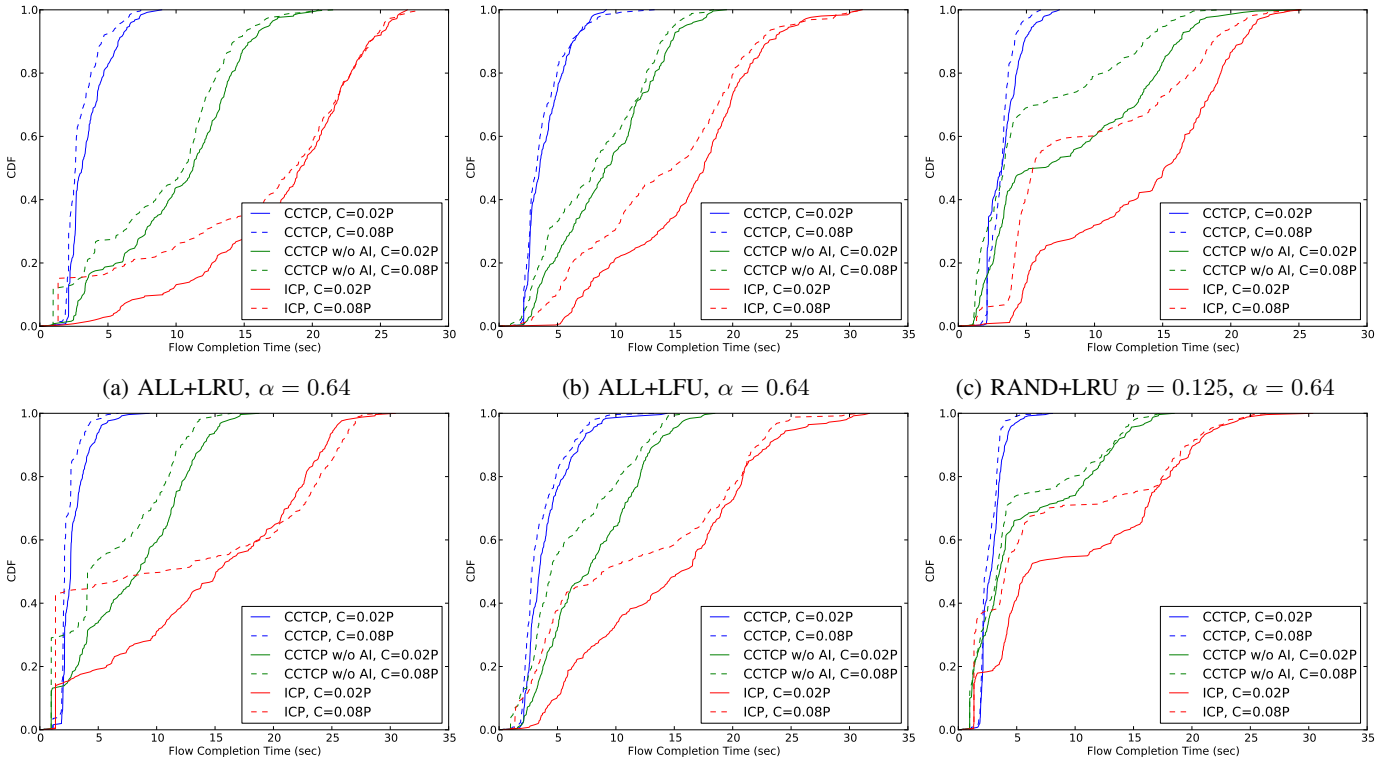
When this takes place and the Data packet is correctly received, the protocol performance does not suffer any degradation. In fact, at the reception of the packet, the receiver analyses the identifier of the router which served the Data packet and updates the CWND and timeout parameters referring to the actual source instead of the expecting one.

If the packet is lost or delayed, instead, the receiver reduces the CWND and increases the timeout of all nodes beyond the expected source, rather then actual source. If the actual loss occurred beyond the expected source, the protocol experiences an unnecessary throughput reduction because the congestion windows of all nodes between the expected source and the congestion point will be unnecessarily reduced. Nevertheless, the system maintains its stability after this perturbation. If on the other hand the packet loss occurs between the expected source and the receiver, the latter might still keep sending Interest packets with an excessive rate. However, our performance evaluation showed that the use of the *virtual packet loss* mechanism described in section II-C mitigates this issue.

III. Performance Evaluation

In this section we present the performance evaluation of CCTCP, carried out through packet-level simulations using the *ns-3* simulator [17] together with the FNSS toolchain [18]. We select Flow Completion Time (FCT) as the key performance metric and analyse it under varying network delays, content popularity distribution, caching policies and cache sizes. CCTCP performance is compared with ICP as well as with a version of CCTCP without *anticipated interests*, which maintains a single CWND and a single timeout value per flow. This allows us to isolate the performance gain provided by the *anticipated interests* mechanism.

The simulation scenario comprises a dumbbell topology composed of five content sources, five content receivers and eight in-network caches. The link capacity is set to 10 Mbps at the edges and 40 Mbps at the core, while all links have a constant delay T_{link} of 5 ms. Content requests have been generated following a Poisson distribution, while content popularity has been modelled using a Zipf distribution with variable α parameter. All content objects have equal size,



(a) ALL+LRU, $\alpha = 0.64$ (b) ALL+LFU, $\alpha = 0.64$ (c) RAND+LRU $p = 0.125$, $\alpha = 0.64$

(d) ALL+LRU, $\alpha = 1.03$ (e) ALL+LFU, $\alpha = 1.03$ (f) RAND+LRU $p = 0.125$, $\alpha = 1.03$

Fig. 4: CDF of Flow Completion Time (FCT) by caching policy, cache sizes and α ($T_{link} = 5ms$)

with each object composed of 1000 chunks of 1 Kbyte. ICP receivers are configured according to the parameters proposed in [10], i.e. $\delta = 0.5$, $\beta = 0.5$ and $\eta = 1$. The initial retransmission timeout values have been set to 3 seconds for all the three protocols evaluated.

The first set of simulations, whose results are reported in fig. 4, compares CCTCP (with and without *anticipated interests*) and ICP under varying conditions of:

- **caching policies**:
  - indiscriminate caching with LRU eviction
  - indiscriminate caching with LFU eviction
  - random caching with LRU eviction (caching probability $p = 0.125$, which results in a content chunk being cached on average once when retrieved from its original source)
- **cache sizes** C: 0.02 and 0.08 times the size of content population P.
- **content popularity distribution:** we use values of 0.64 and 1.03 for the α parameter. They are the minimum and maximum values of α proposed in literature to model the distribution of Internet content popularity.

The results show that, independently of caching policy, CCTCP improves upon ICP and is not sensitive to changes in cache sizes and content popularity skewness (represented by α). Conversely, ICP approaches to some extent CCTCP performance only when the cache size and content popularity increases.

Further analysis shows that the high FCT experienced by ICP flows is mostly caused by frequent window drops triggered by timeout expirations, the majority of which are spurious. This is caused by the timeout algorithm used, which cannot adequately cope with content chunks originated by frequently changing sources. As a consequence, ICP erroneously identifies Data packets served by sources located further away as a precursor of congestion. This phenomenon is exacerbated by rises in cache dynamics, which occur when the cache insertion and eviction rates increase as a consequence, for instance, of smaller caches and content popularity more homogeneously distributed. Under such conditions the ICP performance deteriorates. On the contrary, CCTCP deals effectively with high cache dynamics and achieves lower and more predictable FCT under all the conditions studied. Finally, the performance gain directly traceable to the use of the *anticipated interests* mechanism is substantial. For example, in the case of $\alpha = 0.64$ and ALL+LRU policy (fig 4a), it reduces the mean FCT by 65.2% when $C = 0.02P$ and by 66.9% when $C = 0.08P$.

One situation where ICP approaches the performance offered by CCTCP is when the link delay decreases in comparison to the queueing delay, as shown in fig 5. While reducing link delay T_{link} does not affect CCTCP, ICP performance increases drastically to the point of convergence with CCTCP

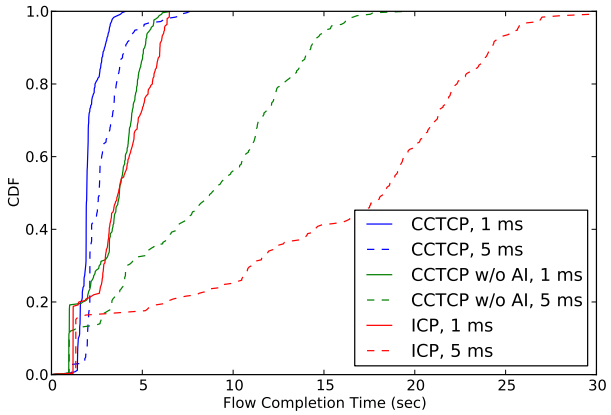


Fig. 5: Impact of link propagation delay (T_{link}) on FCT (caching policy ALL+LRU, $\alpha = 0.8$, $C = 0.04P$)

when no *anticipated interests* are used or when content is retrieved from caches very close to each other. This occurs because reducing propagation delay minimizes variability in the RTT measured by the receiver, which is caused by the retrieval of chunks from different sources. As a result, ICP timeout calculation identifies more accurately congestion events and the occurrence of spurious timeouts is mitigated.

These findings are in line with the assumptions under which Carofiglio *et al.* [10] assessed the performance of ICP, i.e. that propagation delay needs to be negligible in comparison to the queueing delay.

In conclusion, it is important to notice that in all the cases presented above the flows with the lowest FCT consistently achieve an almost identical FCT under either ICP or CCTCP. These flows refer to the downloads of the few top popular contents that are cached almost entirely in the closest cache to the receiver. As a result, all chunks are served by the same node, effectively obviating the need for *anticipated interests*.

IV. Conclusions

This paper presents CCTCP, a scalable receiver-driven transport protocol for CCN. Through the use of a novel *anticipated interests* mechanism, CCTCP is capable of reliably predicting the location of content chunks before they are requested and consequently accurately estimating the retransmission timeout.

Performance is evaluated using *ns-3* simulations and comparisons are drawn with ICP, a key alternative proposal. Results demonstrate that the CCTCP performance is not impacted by variations in caching policy, content popularity distribution, propagation delay or cache sizes, thus validating the use of the *anticipated interests* mechanism to improve performance.

More importantly, CCTCP outperforms ICP in all the scenarios evaluated, especially in all cases where content chunks are scattered along a path, which may be prevalent in the presence of considerable cross traffic or evenly distributed content popularity. The ICP performance gets close to that of CCTCP only when the content is fetched from caches located close to each other and when most of the delay is caused by queueing with no *anticipated interests* used by CCTCP.

To the best of our knowledge, this is the first transport protocol for CCN that effectively addresses the performance issues arising from such content caching dispersion without requiring per-flow state in intervening routers.

Acknowledgment

The research leading to these results was supported by the European Community's FP7 programme under grant agreements ICT-248784 (COMET) and ICT-318708 (C-DAX). The authors alone are responsible for the content of this paper.

References

[1] C. Inc., "Cisco visual networking index: Global mobile data traffic forecast update, 20112016."

[2] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009.

[3] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. de Blas, F. Ramon-Salguero, L. Liang, S. Spirou, A. Beben, and E. Hadjioannou, "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *Communications Magazine, IEEE*, vol. 49, no. 3, pp. 112 –120, march 2011.

[4] N. Fotiou, D. Trossen, and G. Polyzos, "Illustrating a publish-subscribe internet architecture," *Telecommunication Systems*, pp. 1–13, 2011.

[5] P. Poyhonen and O. Strandberg, "D-3.1 the network of information: Architecture and applications," SAIL project, Tech. Rep., 2011.

[6] V. Jacobson, "Congestion avoidance and control," in *Symposium proceedings on Communications architectures and protocols*, ser. SIGCOMM '88. New York, NY, USA: ACM, 1988, pp. 314–329.

[7] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of ccn-caching trees," in *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, ser. NETWORKING'11. Berlin, Heidelberg: Springer-Verlag, 2011.

[8] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, march 2012.

[9] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 55–60.

[10] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp: Design and evaluation of an interest control protocol for content-centric networking," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, march 2012, pp. 304 –309.

[11] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi, "Transport-layer issues in information centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ser. ICN '12. New York, NY, USA: ACM, 2012.

[12] S. Arianfar, P. Nikander, L. Eggert, and J. Ott, "Contug: A receiver-driven transport protocol for content-centric networks," in *IEEE ICNP 2010 (Poster session)*, 2010.

[13] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, march 2012, pp. 322 –327.

[14] G. Carofiglio and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ser. ICN '12. New York, NY, USA: ACM, 2012.

[15] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582 (Proposed Standard), Internet Engineering Task Force, Apr. 2012.

[16] R. Gupta, M. Chen, S. Mccanne, and J. Walrand, "Webtp: A receiver-driven web transport protocol," in Proceedings of IEEE INFOCOM99, Tech. Rep., 1998.

[17] "The ns-3 simulator," http://www.nsnam.org.

[18] L. Saino, C. Cocora, and G. Pavlou, "A toolchain for simplifying network simulation setup," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '13. ICST, Brussels, Belgium, Belgium: ICST, 2013.