Deadline-based Resource Management for Information-Centric Networks

Somaya Arianfar, Pasi Sarolahti, Jörg Ott
Aalto University, Department of Communications and Networking
{Somaya.Arianfar, Pasi.Sarolahti, Jorg.Ott}@aalto.fi

ABSTRACT

Unlike in traditional IP-based end-to-end network sessions, in information-centric networks the data source may change during a communication session. Therefore the response time to subsequent data requests may vary significantly depending on whether data comes from nearby cache, or a distant source. This is a complication for designing resource management, reliability and other algorithms, that traditionally use RTT measurements for determining when data is considered lost and should be retransmitted (along with related congestion control adjustments). This paper discusses a different approach for designing resource management in information-centric networks: data packets are assigned with a lifetime, that is used as a basis for scheduling and resource management in the network, and for congestion control and retransmission logic at the end hosts. We demonstrate an initial evaluation of this approach based on ns-3 simulations on CCN framework.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

Content-oriented Networking, Resource Management

1. INTRODUCTION

In the traditional network transport round-trip time measurement plays an important role. Retransmission timers are typically set based on round-trip time (RTT), with the aim of optimizing rapid enough retransmission time for potentially lost packets, while avoiding unnecessarily retransmitting data that is still under delivery. For window-based protocols such as TCP, round-trip time is also a factor that affects the rate of congestion control adjustments, and ultimately the transmission performance [11, 15].

Algorithms based on round-trip time measurement come with an implicit assumption that the end hosts of the communication session remain the same. Since many information-centric networking models do not assume fixed communication end points, but rather enable distributed delivery of data from multiple potential sources, RTT measurement becomes a nearly useless concept: even if we know RTT distribution (delay between request and corresponding response) for past data, we cannot know whether that matches the RTTs of upcoming data, because the data source may change at any time [2, 3]. This makes data retransmissions difficult, because end host receiver cannot know when to re-request data. The impact of round-trip time is not limited to the behavior at the end host. In some cases also router algorithms either directly or indirectly depend on the assumptions of the typical RTTs. RCP [6] is one example of such algorithm in IP network, while an ICN equivalent of it could be found in different interest shaping algorithms such as the one in [18].

Another aspect that complicates RTT measurement – not only in information-centric networks, but also in general – is the extent of buffering in today's networks. Excess buffering (sometimes termed "bufferbloat") significantly impairs the communication performance has been observed in some networks [13]. Most proposed Information-centric network architectures come with a built-in capability storing data along the communication path, and are thought to have a large capacity for storing data for later transmission. This, of course, does not make RTT estimation any easier.

Because of the inherent difficulty of path RTT estimation in information-centric networks we propose an alternative way to approach data retransmission and resource management in general: each network packet is supplied with a *lifetime* that tells how long it lives in the network before being dropped. The lifetime is used primarily in three ways in our work: first, from lifetime the data receiver knows how long to wait for answer before re-requesting data. Second, the network routers can use the lifetime as part of their queue management policy: it is ok to delay data that does not have tight lifetime constraints, if that helps to provide better service for more delay-sensitive data. We also aim at discarding packets as soon as it becomes obvious that they are going to miss their deadline, and thus make space for more useful data as soon as possible. Third, we use the lifetime information in interest packets to predictively discard them as a resource management measure, if it seems that the data would not reach its destination in time. This release resources early for other traffic with better chance of survival.

2. BACKGROUND

In the following we discuss the related work on assigning data lifetimes for traffic management, and resource management in information-centric networks in general.

2.1 Lifetimes

The concept of lifetime is not new in networking: the IPv4 pro-

ICN'13, August 12, 2013, Hong Kong, China.

tocol has "Time To Live (TTL)" field that originally indicated the number of seconds a packet is allowed to remain in the network [17]. Because the actual packet propagation times are typically much less than a second, and on the other hand, all forwarding nodes are required to decrement this field at least by one, the actual meaning of the TTL field has evolved into hop count rather than real lifetime.

Packet lifetimes have also been discussed in the context of real-time transfers over cellular networks [9]. We extend this thinking into information-centric networks and resource management in general. The notion of data lifetimes has also been applied in Delay-Tolerant Networks [7]. However, while Delay-Tolerant Networks are designed to operate on relatively long time scales, we assume lifetimes that are on the order of few round-trip times, i.e., no more than a few seconds in typical networks.

The notion of lifetime has not been discussed in the context of information-centric networks extensively, although cached data at intermediaries, or the pending data subscriptions could be assigned with a maximum lifetime. We are not aware of previous work for applying packet-level lifetimes for resource management in information-centric networks, as we do in this paper.

2.2 Network Rate Control

A few rate control algorithms have been proposed for information-centric networks. Typically these follow the principle of additive increase, multiplicative decrease (AIMD) that is familiar from TCP [4, 1], but at the receiver, regulating the rate of data requests. Also interest shaping at the intermediate nodes has been proposed [18, 5]. These algorithms are partially based on RTT estimates between the request and corresponding data. However, we argue that in information-centric networks, where data can be sent by any of the potentially large number of sources or network caches, measuring RTTs is not useful because of its potentially large variance. Therefore, in the following we propose an approach that is independent of RTT estimation.

3. NETWORK MODEL

We now focus on the basic communication model of lifetime based packet delivery: packets with the remaining lifetime information in their headers, and the queuing models that distinguish these packet lifetimes in the transmission scheduling. We assume receiver-oriented information-centric communication model where data is requested by the receiver, and may be sent by any number of possible alternative sources or network caches.

3.1 Packets with lifetimes

In our model each network packet is provided with a lifetime that is specified on the granularity of milliseconds. The sender initializes the packet lifetime primarily based on the application requirements indicated to the networking implementation. For example, a conversational application might request a packet lifetime of 100 milliseconds, whereas for batch file transfer significantly longer lifetimes would be sufficient. After the lifetime for packets has been specified, the receiver knows how long it must wait before declaring the packet lost, and re-request the data if needed. There is a trade-off that an implementation needs to consider when choosing the packet lifetimes: too long lifetime makes loss recovery more inefficient. Too short lifetime may increase the likelihood of premature elimination of a packet. An implementation may also adjust the lifetime of consecutive packets adaptively based on some heuristic. Unlike with TCP and similar unicast protocols, detecting packet loss from out-of-order acknowledgments is not possible, because in ICN there may be multiple data sources and packet reordering may therefore be an intrinsic behavior. Similarly, taking RTT samples is not useful, as the expected RTTs may vary significantly between packets. Therefore lifetime is the main recovery method in our model.

The underlying network characteristics should not be a factor affecting the choice of the lifetime, but the lifetime should be set solely based on the application requirements. Similarly, the lifetime should not be directly adapted based on the observed network characteristics such as RTT or packet loss rate, without involving the application. However, an application can adapt its behavior based on observed network conditions in a way that may result also in packet lifetime changes, for example by choosing a different content encoding.

A separation should be made between the packet lifetimes and lifetimes used for cached data in the network. Packet lifetimes are primarily intended for the uses for protocol operation, and are thought to be on the order of few seconds at maximum. The data receiver specifies the packet lifetime based on how long it is ready to wait for response. On the other hand, data source can specify the time how long data is valid to be cached, to be shared between multiple receivers. This time is typically longer, for example on the order of minutes or hours.

Packet expiration can be indicated in packets in two ways: either as a absolute "wallclock time" indicating the deadline after which the packet should be discarded, or as a relative remaining lifetime, that should be decremented at each hop[1]. Both approaches have their challenges: for reliably decrementing remaining lifetime at each hop, a router should know how long the packet spends in router processing and queues, which is doable, and the time the packet takes to propagate over a link, which may be challenging, for example on wireless links, or multi-access links in general. Indicating a deadline would require synchronized clocks, or that nodes make adjustments on the deadline based on what they know about the clock differential to the neighbors. We choose to follow the deadline approach, because the time difference between neighboring hosts can be estimated with sufficient detail by having the hosts mutually exchange their local view of the clock, and compensating the communication propagation time based on RTT measurements between neighbors. The Network Time Protocol (NTP) [14] applies similar methods in a rather successful way. After such adjustments are made, we believe that the deadline can be managed with sufficient accuracy to allow correct operation, as long as each hop along the path makes similar adjustments. Global synchronization of clocks is not required for correct interpretation of lifetimes, as long as each node makes the needed adjustments on the deadline communicated on wire based on observed clock differences between neighbors. There may be inaccuracies in this estimation, but as long as the inaccuracies remain relatively small, they do not prevent correct protocol operation.

When packet arrives at a node, the node checks its deadline from the header. The header is indicated relative to the source node's clock, so the receiving node adjusts the deadline based on the measured time difference to the source node ($Diff_i$, where i identifies one of the neighboring hosts). If the packet is not expired, it is scheduled for forwarding.

3.2 Queues with lifetimes

Even though some of the ICN research appears to assume that network queues are not needed with ICN because of the network-based content storage model [12], we believe that for scheduling purposes queues are inevitable in any network, including information-

[1]In the text we use the terms *lifetime* and *deadline* interchangeably. However, our implementation assumes deadlines as absolute timestamps.

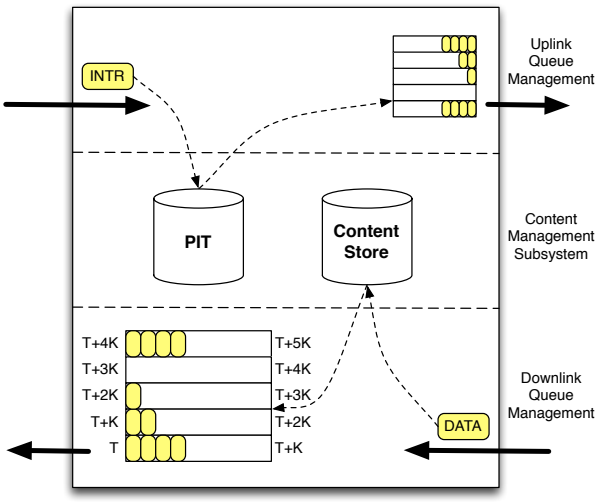

Figure 1: Queuing model for lifetimes.

centric networks. Data can be sent out on an interface only one packet at a time, and if the demand for outgoing data exceeds the transmission capacity, queues are needed for scheduling the outgoing packets, regardless of the data storage model.

The queue management at routers can leverage the lifetime information in packets to enable differential treatment for urgent traffic. Various queuing models could be applied, and for example the *Shortest Remaining Processing Time (SRPT)* queuing has been proposed for web transfers [10]. However, in the following we discuss a simple deadline-based queuing method, that we believe to be more appropriate for packet-level information-centric networking.

Figure 1 illustrates our assumed network model. We make a distinction between content management functions, such as the Pending Interest Table (PIT) and Content Store in CCN, and between the queue management functions that are used for scheduling transfers to outgoing links, and if needed, for discarding packets that are deemed not to be useful considering their remaining lifetime.

The router queue space is divided into N different subqueues to which packets are assigned based on their deadlines. The illustrated group of time-limited queues is called a *queue stack*. In the figure T indicates the starting time of the first queue that contains packets with deadlines between time T and T+K, the second queue contains packets that have deadlines between T+K and T+2K, and so on. At any moment, the current time is between T and T+K, and when time reaches T+K, the first queue, and any possible unsent packets in the queue are discarded, because the remaining packets have missed their deadline. In addition, a new queue is allocated at the end of the queue stack. There is a separate queue stack for each outgoing interface of a node.

A network node can choose its queue management algorithm, and parameters N and K independently of other nodes. The subqueues of a queue stack do not have fixed length limit, but rather their length is limited based on the measured outgoing bandwidth and the length of the time window (K) per subqueue. For example, if K is 100 milliseconds, and outgoing bandwidth is 1 Mbps, storing more than 12500 bytes worth of packets to the first queue is not useful, because further packets would be missing their deadline. The following subqueue can have a similar limit to the amount of data, but the potential maximum length can be larger for each subsequent queue, if the prior queues do not use all of their associated time slot.

It is not in receiver's interest to assign only short lifetimes to packets because of the combined effect of two properties. First, the queue stack is in constant move: when a queue expires from the head of the stack, it is removed, and a new queue is added to the end of the stack. As this rotation happens, packets that have waited longer in the queue (i.e., they were originally assigned longer lifetimes) move towards the head of the queue stack. Second, when a new packet arrives, it is placed at the end of the particular queue chosen based on the packet deadline. Because of the queue rotation, packets that have arrived earlier (possibly with longer initial lifetime) occupy the first positions of the queue. Therefore, using appropriately scaled lifetimes improves the likelihood that packet will be delivered in time, even if the average delivery delay might increase under increased load. Using excessively short lifetime would cause increased likelihood of packet expiration when multiple packets arrive at the same queue in the queue stack, possibly occupied by earlier arrived packets. By distributing the lifetimes more evenly such that they are uniformly assigned to different queues in queue stack is expected to yield better packet delivery rates. Therefore, we believe that the receiver has a real interest in trying to assign appropriate deadlines for packets.

It is important to distinguish the earlier queue management proposals and QoS mechanisms [8, 16] from the approach described above. The queues in queue stack do not reflect priorities of traffic, and are not flow-aware: each packet is processed independently, which is appropriate for ICN that do not have clearly defined flows.

3.3 Network resource management

With the additional knowledge of packet deadlines, the network nodes can perform advanced scheduling algorithms for more efficient resource management. Before acting on a data request (e.g., Interest packet in CCN), the router checks the current load on the return path, and calculates whether it is possible to deliver the response within the assigned deadline, considering the currently scheduled data transmissions. If it seems that the data cannot be delivered by the deadline, the data request is dropped immediately. This way the router predictively avoids using the upstream network resources, and helps in reducing load on the return path.

Our scheduling algorithm is based on a few assumptions, similar to the ones made in CCN [12]. In the following we also use the terminology and communication model from CCN: the packets for requesting data are called "Interest" packets, which are responded by "Data" packets, and for each data packet a separate Interest packet is required. We assume that the paths are symmetric: data follows the reverse path of the Interest packets, and the packet lifetime (or deadline) is set by the sender of the Interest packet. When Data packet is sent in response to the interest, it inherits the lifetime from the Interest packet.

When an Interest packet arrives at router, it evaluates the status of queue stack both on the next hop interface, to see if the Interest can be forwarded in time; and on the previous hop interface, to estimate whether the returning data packet can be delivered in time. If either of the tests indicate that the data cannot be delivered in time, the Interest packet is discarded, because it would only unnecessarily consume the upstream network resources if forwarded. In both above cases the router calculates how many packets are scheduled before the current Interest on the respective queue stack, and what is the estimated queue delay based on the knowledge of the outgoing link bandwidth. For the return path queue the router makes an *pre-allocation* for the expected upcoming Data packet. This way the router avoids over-subscribing the return path: the Interest is forwarded only if the router calculates it can deliver the data packet. The pre-allocation is replaced with actual Data packet, when it arrives at the router. Note that even if the router determines

that it can deliver the Data packet at the Interest time, the Data packet can still be late, because the upstream round-trip time is not known. However, this mechanism provides a best-effort way to predictively reduce congestion at the time of receiving the Interest packet.

If data is in the content store of the local node, the queue stack for forwarding the Interest does not need to evaluated, and the pre-allocation is not needed. In that case the data is just queued for transmission on the return path queue stack, based on the deadline received on arriving interest.

Algorithm 1 details this predictive interest discard approach in more specific terms, for an arriving Interest packet with deadline D. Q_i and Q_d are the queue stacks for interface where the Interest packet is forwarded next, and where the arriving Data packet is expected to be delivered, respectively. q_i and q_d represent particular queues in these stack, as determined by the deadline in the packet. $startT_q$ and $endT_q$ represent the starting and ending time boundaries in these queues, and $currentT$ stands for the current time. $threshold_i$ and $threshold_d$ are the calculated maximum amount of data that can be delivered to the link before the deadline, considering outgoing link bandwidth (B_i, B_d). J_i and J_d are subsets of the respective queue stacks to represent queues from which packets are sent before the newly arriving packet, and S_i and S_d are the amount of data currently in these queues that will be sent before the newly arrived interest packet (S_i), or the data packet expected to arrive (S_d). $length(s)$ refers to the current length of the queue s in bytes). If S exceeds the calculated threshold value in either direction, we know that data cannot reach the receiver in time, and the interest can be discarded. We also note that $endT$ in the last queue of the queue stack is infinite, to cover packets with unexpectedly long lifetime.

$Q_i = \{Queue\ stack\ for\ Interest\}$;
$Q_d = \{Queue\ stack\ for\ expected\ Data\}$;
Pick $q_i \in Q_i$ so that $D \in [startT_{q_i}, endT_{q_i}[$;
Pick $q_d \in Q_d$ so that $D \in [startT_{q_d}, endT_{q_d}[$;
$threshold_i = B_i * (endT_{q_i} - currentT)$;
$threshold_d = B_d * (endT_{q_d} - currentT)$;
$J_i = \cup q_j, \forall q_j \in Q_i$, so that $startT_{q_j} <= startT_{q_i}$;
$J_d = \cup q_j, \forall q_j \in Q_d$, so that $startT_{q_j} <= startT_{q_d}$;
$S_i = \sum_{s \in J_i} length(s)$;
$S_d = \sum_{s \in J_d} (length(s) + PreAllocated_s)$;
if $S_i >= threshold_i$ **then**
  Discard packet;
else
  if $S_d >= threshold_d$ **then**
    Discard packet;
  else
    Forward packet to q_i;
    $PreAllocated_{q_d} += Max\ packet\ size$;
  end
end

Algorithm 1: Predictive interest filtering

Later, when data packet corresponding to the interest arrives, it is placed on queue q_d based on the deadline that should remain unchanged from the deadline of the corresponding interest packet. The value of $PreAllocated_{q_d}$ is reduced by maximum packet size to keep the bookkeeping in balance. Naturally, when a queue expires, i.e., $endT_q$ becomes less than current time, the pre-allocations on that queue are forgot at the same time as the packets still in the queue are discarded.

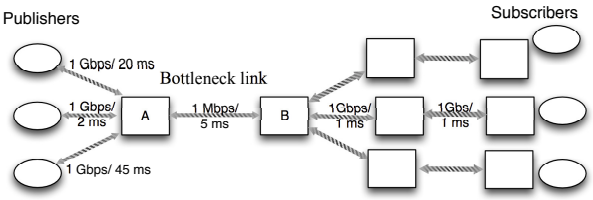

Figure 2: The simulation topology

It is possible that the Interest packets are forwarded to more than one interface. In such case, Algorithm 1 is evaluated separately for each outgoing interface before the packet is forwarded to that interface.

3.4 Receiver Behavior

Because in an information-centric network the communication is typically receiver driven, potentially multi-party interaction, TCP is not a suitable protocol to be used as transport. This paper does not specify any transport protocol in detail, but in the following we describe our assumptions of the data transmission logic at the receiver, for example as applied on top of the CCN architecture.

The data receiver dynamically adjusts the rate at which it sends Interest packets to the network, responding to packet losses or other congestion notifications following an additive increase, multiplicative decrease congestion control principle similar to TCP. If the receiver does not receive an expected Data packet in the specified lifetime, the network has dropped the expired packet. After the lifetime is expired, the receiver does two things: first, it reduces the rate of sending interests as a congestion reaction, because the failure to meet the deadline can be assumed to be caused because of excess queuing in the network. Second, the receiver may optionally re-request the data, if reliable transmission mode is desired. In our implementation we assume similar behavior to TCP: the rate of outgoing interests is increased steadily, and a packet loss triggers dropping the sending rate to half of the previous rate. Adjusting lifetime is not considered to be a proper congestion reaction action, because – as discussed earlier – the lifetime should primary be based on the properties of application. If an application can operate with long lifetimes, it should use such lifetimes from the beginning, and not only after congestion events in the network. It should be noted that long lifetime does not automatically result in long delay, unless the network is under heavy load.

4. EVALUATION

We analyzed the basic dynamics of the deadline-based resource management model using the ndnSIM simulation framework for ns-3[2], conducting simulations on a dumbbell topology illustrated in Figure 2. In this topology, all the links between different subscribers and the router *B* share the same attributes: 1 Gbps capacity and 1 ms propagation delay. The round-trip delay between data sources and receivers ranges from a few milliseconds to 100 ms, excluding the possible effects of queuing. We use ndnSIM's normal communication model of Interest packets consisting of 32 bytes (for content name), and data packets being 1055 bytes. In our simulation the data flow is unidirectional, as illustrated in Fig. 2.

We start our evaluation by analyzing different RTTs and deadline settings for continuos ongoing content transfer operations in the

[2]`http://ndnsim.net/`

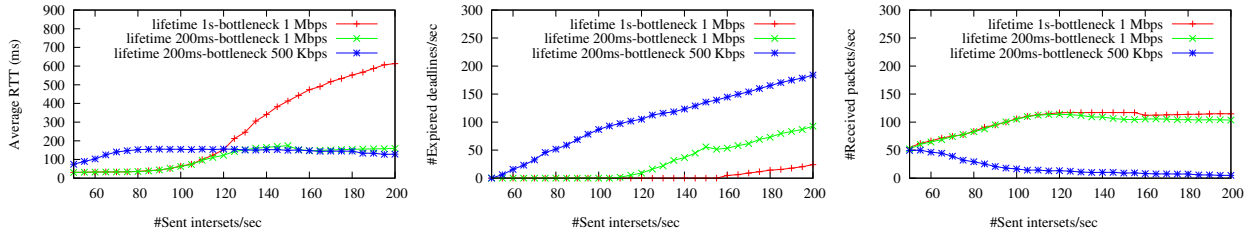

Figure 3: The variation in the (a) average RTT on the left, (b) number of dropped packets in the middle, (c) number of received packets on the right

network. Figure 4 compares the number of received and dropped packets and the average experienced RTT in different scenarios. We have chosen 3 different cases in which lifetime is set either to 200 milliseconds or 1 second, while the bottleneck link capacity is set to either 1 Mbps or 500 Kbps. Figure 3a illustrates the simple observation that depending on the deadline setting, content transfer operations can tolerate different range of RTTs. In our simulation scenario setting the lifetime to 200 ms keeps the average experienced RTT below 200 ms. In the same scenario, choosing the lifetime of 1s allows for a wider range of RTT variation from a few milliseconds of two-way propagation delay to about 700 milliseconds, involving a significant amount of queue delay. Figure 3b illustrates that the number of dropped packets increases as the deadline is set to the lower values. In the same network setting, having the lifetime set to 1 second allows lower drop rates compared to the case of packets with 200 millisecond lifetime. Therefore, if shorter lifetimes are desirable, it is useful for the application to limit its data request rate to reduce the likelihood of packet losses. Figure 3c shows that if there is sufficient network capacity, the deadline setting does not have significant effect on the transmission performance: with the bottleneck of 1 Mbps the performance is the same both with 200 ms and 1000 ms lifetimes. Figures 3b and 3c also show that if the network is congested, the number of received packets decreases as the number of packet losses increases (see 200 ms lifetime with 500-Kbps bottleneck link). If no resource control is applied either from the network side or from the application side, in extreme cases hardly any data gets through in the allowed lifetime.

We next analyze different scheduling models that can help the network to better meet the assigned deadlines. For this purpose we use three different kinds of traffic models in our simulation scenario. Type A traffic has 1000 ms packet lifetime with retransmissions for missing packets (for example, time-insensitive reliable file transfer); type B with 200 ms lifetime with retransmissions, and type C with 200 ms lifetime, but without retransmissions (for example, real-time media streaming with tighter timing constraints).

The routers in our simulation model have a queue stack of 15 queues at each interface with 100 ms time window for each queue, i.e., the last queue accepts packets arriving with more than 1400 ms remaining lifetime. The router applies early discard of packets before the queues are fully used, to prevent bursts of traffic from occupying full queue capacity in a queue stack, and thereby leaving a window of opportunity for urgent traffic that may arrive later.

We analyze three different scheduling heuristics. First is a traditional single queue FIFO, where packets are forwarded in the order they arrive. The FIFO scheduling has one element per content item: if another Interest comes for a same name that already exists in the queue, it shares the queue position with the previous request. Second scheduling approach is Last-In-First-Out with a single queue, which we expect to yield different results especially for data with short lifetimes. Finally, we analyze the deadline-based scheduling as described in Section 3.3.

Figure 4 shows the goodput with the three scheduling heuristics, when three different types of content flows are in transit (as described above) as the intensity of the load on the network increases. The FIFO scheduling results in devastating effect on the content with short lifetimes, because as the queues build up on higher load, these packets have no chance of surviving. In contrast, with LIFO scheduling the different traffic classes are treated more equally. The deadline-based scheduling gives most capacity on the packets with short lifetimes, but less capacity for the traffic with longer lifetimes. We can also see that under high load, the performance suffers with the single queue models, leading to unpredictable behavior.

For FIFO scheduling there is an interesting difference between the reliable and unreliable variants of traffic with short lifetimes. This is a result of two interacting details in our implementation: following the information-centric principles, the Interests of the same data share a single queue position, instead of taking multiple positions in the queue. On the other hand, whether to discard or send a packet is evaluated only at the head of the queue. With long queue there will be expired elements waiting for their turn, but because receiver re-sends the Interest, they occupy the earlier position. This works in favor of the reliable traffic in the FIFO scheduling model.

Figures 5a, 5b, and 5c illustrate the number of discarded data packets for each traffic type. They explain why in case of FIFO the performance of unreliable, short-lifetime traffic suffers: the packets get delayed and dropped by network queuing. The other scheduling heuristics treat the different traffic types more equally.

5. CONCLUSIONS

In this paper we described a new approach to network resource management, leveraging packet lifetimes. This approach makes the network management independent of round-trip time measurement, which is useful property in environments where RTT measurement is difficult, such as information-centric networks. We presented a scheduling algorithm that makes use of the packet lifetimes, trying to ensure that packets are delivered within their assigned lifetimes, and compared it with alternative scheduling algorithms based on ns-3 simulations.

Even though this work was based on information-centric networks, deadline-based network operations may be useful also in other contexts with challenged RTT measurement. For example, bufferbloat has been identified as a significant challenge for the resource management in IP networks, because of the varying packet delays. Similar problem has also been observed on wireless links, where unreliable link technology and link-layer reliability mechanisms cause high variations in packet transmission delays. We will continue to investigate these avenues further, along with a

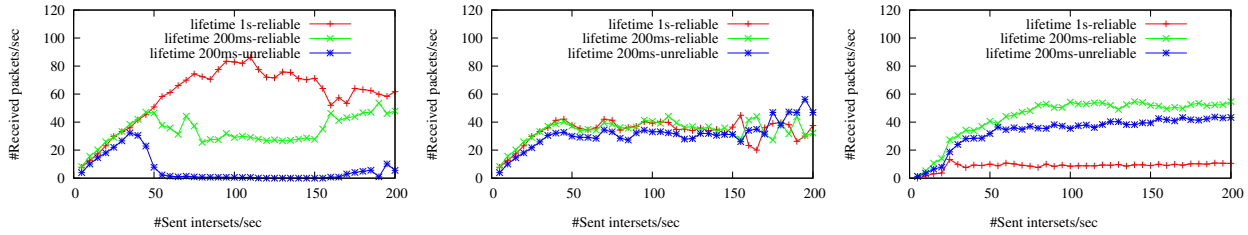

Figure 4: Number of received packets (a) with FIFO style scheduling on the left, (b) with LIFO style scheduling in the middle, (c) with time scheduled queues on the right.

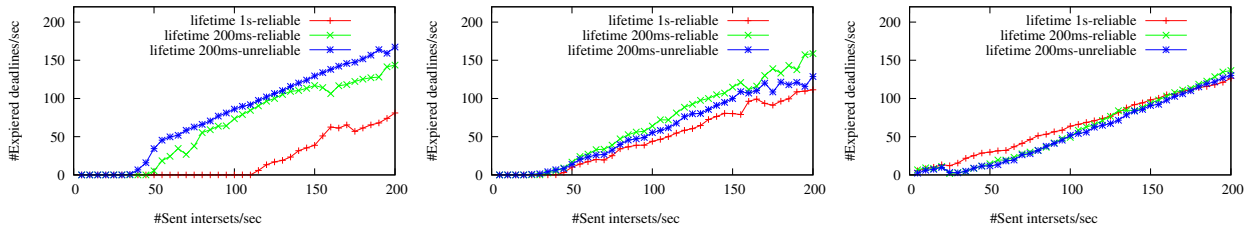

Figure 5: Number of expired interests (a) with FIFO style scheduling on the left, (b) with LIFO style scheduling in the middle, (c) with time scheduled queues.

more thorough performance evaluation and further development of scheduling algorithms for information-centric networks.

Acknowledgments

This research was partially funded by the EU FP7 PURSUIT project (FP7-INFSO-ICT-257217).

6. REFERENCES

[1] S. Arianfar, L. Eggert, P. Nikander, J. Ott, and W. Wong. Contug: A receiver-driven transport protocol for content-centric networks. Technical report, PURSUIT publish-subscribe project, 2011.

[2] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, pages 5:1–5:6, New York, NY, USA, 2010. ACM.

[3] S. Arianfar, P. Sarolahti, and J. Ott. Reducing server and network load with shared buffering. In *Proceedings of the 2012 ACM workshop on Capacity sharing*, CSWS '12, pages 33–38, New York, NY, USA, 2012. ACM.

[4] G. Carofiglio, M. Gallo, and L. Muscariello. ICP: Design and evaluation of an interest control protocol for content-centric networking. In *IEEE NOMEN (INFOCOM WKSHPS)*, 2012.

[5] G. Carofiglio, M. Gallo, and L. Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *Proc. of ACM SIGCOMM workshop on Information-Centric Networks (ICN)*, 2012.

[6] N. Dukkipati. *Rate Control Protocol (RCP): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008.

[7] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proc. of ACM SIGCOMM 2003*, pages 27–34, Karlsruhe, Germany, Aug. 2003.

[8] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *INFOCOM '94. IEEE*, pages 636–646 vol.2, 1994.

[9] A. Gurtov and R. Ludwig. Lifetime packet discard for efficient real-time transport over cellular links. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(4):32–45, Oct. 2003.

[10] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2):207–233, May 2003.

[11] V. Jacobson. Congestion avoidance and control. In *Proc. of ACM SIGCOMM*, Stanford, CA, USA, 1988.

[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *Proc. ACM CoNEXT*, 2009.

[13] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: illuminating the edge network. In *Proc. of ACM Internet Measurement Conference (IMC) '10*, IMC '10, pages 246–259, Melbourne, Australia, 2010. ACM.

[14] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905, June 2010.

[15] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proc. of ACM SIGCOMM '98*, pages 303–314, Vancouver, Canada, September 1998.

[16] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, June 1993.

[17] J. Postel. Internet Protocol. RFC 791, Sept. 1981.

[18] N. Rozhnova and S. Fdida. An effective hop-by-hop interest shaping mechanism for ccn communications. In *IEEE NOMEN'12 Workshop*, pages 322–327, 2012.