Pub/Sub-Sum: Content-summarization-based Pub/Sub Protocol on Information-centric Networks

Jongdeog Lee, Suk Min Hwang, Tarek Abdelzaher
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL
{jlee700, shwang53, zaher}@illinois.edu

Kelvin Marcus, Kevin Chan
Computer and Information Science Directorate
US Army Research Labs
Adelphi, MD
{kelvin.m.marcus.civ, kevin.s.chan.civ}@mail.mil

***Abstract*—In the age data overload and scenarios that require fast distributed situation understanding, we envision that content summarization services will become a critical capability of the underlying networked systems. Previous work, called InfoMax, proposed such a service in the transport layer to minimize semantic redundancy of transmitted content and maximize information coverage. We extend this work in three dimensions. First, we adapt summarization to the needs of *streaming* content and develop a corresponding publish-subscribe protocol with on-the-fly extractive summarization (Pub/Sub-Sum) of continuous content streams (as opposed to extractive summarization of fixed data sets). Next, we support many-to-many communication between publishers and subscribers, as opposed to InfoMax, which is designed to disseminate data from one producer to multiple consumers. Lastly, we introduce a new type of congestion handling mechanism that adaptively controls the level of summarization by considering available network bandwidth. We conducted experiments for functionality and performance on Mininet (a network emulator) and on a real device testbed. Evaluation results indicate that the new protocol summarizes data appropriately to available network resources, offering an improved compromise between received data quality and resource consumption.**

***Index Terms*—pub-sub protocol, information-centric network (ICN), data summarization, congestion control**

I. Introduction

Our work is motivated by the need to enable better team situation understanding in dynamic scenarios involving large data streams originating from multiple sources that contain information critical to the team. An example might be a disaster response team that aims to keep up with an evolving threat, such as a flood, via multiple camera feeds from the affected neighborhood(s). Arbitrary contention among different data streams may result in unpredictable consequences such as disproportionate data loss in streams transmitted over longer distances or in streams originating across tighter bottlenecks. In such scenarios, the underlying network services must be able to summarize content in a manner that provides a more balanced situation understanding.

There exist broadly two techniques for data summarization: namely, *abstractive* [1] and *extractive* [2]. The former technique aims to "understand" the data and report an abstraction that summarizes the key parts. An example would be to paraphrase the gist of a longer text article. The latter technique extracts representative data samples, such as video clips that summarize the most important moments of a longer event (such as a sports game). We adopt extractive summarization in the current paper due to its relative simplicity (being a data sampling/selection problem) and independence from the details of the data format itself.

Our work extends InfoMax, proposed by a subset of the authors [3], [4] to summarize static data sets from a single producer. Infomax operates on top of information-centric networks (ICNs), where data has names that can be resolved for query routing purposes [5]. It adopts a hierarchical name-space. Accordingly, it logically organizes (i.e., names) data in a hierarchical manner such that higher content similarity is translated into a longer shared name prefix. The run-time protocol then prioritizes data so that the next data item to transmit is semantically less redundant with the previously transmitted data based on this simple naming convention. More specifically, it transmits the data item with the smallest shared name prefix (with previously transmitted items) next. The protocol is implemented on a named-data network (NDN) stack. NDN is a representative of ICNs [6].

Our work overcomes two key InfoMax limitations. The first limitation of InfoMax is that it is strictly a one-to-many summarization scheme. Only one data source is assumed. Redundancy among multiple sources (e.g., cameras monitoring the same building from overlapping viewpoints) is not addressed. The second limitation is that InfoMax targets static (or slowly evolving) content, such as images of key tourist landmarks or social media feeds that are aggregated into static bins (say, every 6 hours) so they can be statically summarized. This limitation is problematic especially for streaming data such as video. The higher tempo involved in continuous video summarization precludes solutions that work on an individual object (e.g., frame) level. Instead, more efficient solutions must approach summarization at a higher level, such as the level of sensor selection (e.g., selection of cameras that have the best and most complementary vantage points at a given time).

Pub/Sub-Sum handles both of the above limitations. First, by thinking of the content name space as a space for organizing named data *streams* (not individual objects), we are able to choose a subset of streams to share in a way that maximizes a notion of situation coverage. Second, we extend the approach to handle the multi-producer case. To this end, locally registered stream prefixes are propagated across the network to share prefix information among the nodes. The prefix propagation can be done by the routing protocol or a separate application. In the context of NDN, named-data link state routing protocol (NLSR) [7] or automatic prefix propagation [8] can perform the described job. After getting the prefix information from the network, the protocol constructs a current prefix tree that then constitutes the basis for summarization.

Finally, in moving from static data sets to streams, a new problem is introduced; namely, one of congestion handling. Since the summarization service persists over the duration of the stream, one must adapt the subset of streams that constitute the summary in a manner that is responsive to the current level of network resource availability. Traditional congestion handling mechanisms such as TCP control the sending rate of individual sources without impacting what content to send. We suggest a complementary congestion control algorithm on top of TCP that adapts the level of summarization, which translates into selection of one or more sources to be part of the current summary. Other Pub/Sub-Sum nodes back-off. The algorithm assumes that Pub/Sub-Sum nodes implicitly agree to be cooperative as might be the case with first responders trying to maximize their overall situation understanding.

We first evaluate the protocol in the Mininet simulator by comparing it to a simple baseline with no summarization cabaility. The evaluation results indicate that Pub/Sub-Sum retrieves more balanced content than the baseline, thus increasing situation understanding. We also evaluate the protocol on a Raspberry PI testbed. Testbed experiments confirm that Pub/Sub-Sum functionally works on real devices and the results match with the Mininet simulations.

The rest of this paper is organized as follows. Section II first describes design considerations and decisions, followed by the implementation details in Section III. In Section IV, the Mininet simulation setup and results are provided along with the real testbed experiments. Section V reviews the literature and Section VI concludes this paper.

II. System Design

This section describes the protocol design. There are three design considerations. First, we discuss how to handle streaming content. Second, we present our algorithm for handling multiple sources. Lastly, we describe how to manage congestion in the pub-sub protocol, jointly with the underlying system's congestion handling mechanisms.

A. Handling Streaming Content

Prior work on network services for content summarization assume the existence of a relatively static set of objects that need to be shared. For example, InfoMax [3] prioritizes all data and generates a single list of available named objects that can be broken hierarchically by topic and subtopic. Let us call it the *name catalogue*. Consumers request a piece of the name catalogue on a given topic/subtopic, called *page*, containing a fixed number of names. Once a consumer receives all content of the retrieved page, it may request the next page if more detail is needed. In this manner, consumers can control the amount of content retrieved until their summarization needs are met.

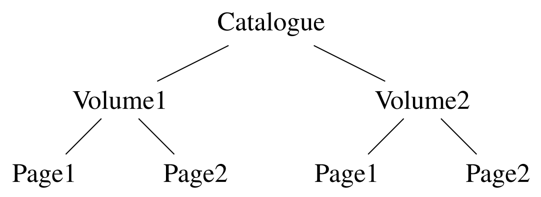


Fig. 1: An example of name catalogue tree with 2 volumes having 2 pages for each

The problem with this method is that the name catalogue is difficult to update once published. This complicates support for real-time streaming applications, where new objects are constantly generated and where summarization has to be done on current (as opposed to archived) content.

To remedy this problem, we focus on individual *streams* as the units comprising the summary. The name space breaks down the set of available streams in a hierarchical manner such that more redundant streams (e.g., feeds from overlapping cameras) share a longer name prefix. Given a query on a specific topic (expressed by a prefix in the same space), the tree of streams under that prefix is summarized by considering only a subset of its branches in the summary; namely, the least redundant branches (i.e., branches whose names share the least prefix). As we show later, the size of the subset is controlled depending on network conditions. Hence, in effect, subscribers subscribe to streams from a different number of branches in the query range that jointly offer the best coverage of that range (for the selected number of streams).

In addition, each publisher periodically creates a sub-list, called *volume*. An example of a resulting catalogue can be found in Figure 1. A name catalogue consists of several volumes containing different content streams. As before, each volume has pages of contents names, where each page has a fixed number of names. The update period decides the number of volumes and pages. Thus, it has a trade-off between update frequency and overhead. The shorter the period, the quicker the update, but the higher communication overhead.

Let's assume that the publisher produces content every second for n seconds. Then the total number of volumes would be $\frac{n}{p}$, where p is the update period. If p becomes smaller, the publisher generates more volumes that contain fewer pages. Otherwise, there will be fewer volumes having more pages. For instance, the publisher may want to have smaller p for the faster update in the example of live video streaming. On the other hand, if a pre-recorded video needs to be streamed out, p can be large since the publisher knows the whole size of the video ahead of time.

B. Supporting Multiple Sources

Our algorithm supports multiple sources by allowing them to exchange name prefixes of data they carry. If an NDN-like

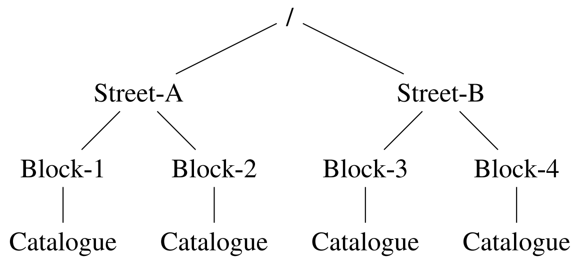


Fig. 2: An example of the prefix tree

routing protocol is running, it may exchange locally registered prefix information. Specifically, in the context of NDN, the named-data link state routing protocol (NLSR) should be able to perform this function [7]. NDN also provides an automatic prefix propagation method, as described in [8]. Otherwise, an application-layer prefix broadcast can do the job.

Once the prefix information is shared, a subscriber can construct prefix (or topic) tree as shown in Figure 2. The example shows four publishers, whose names are /Street-A/Block-1, /Street-A/Block-2, /Street-B/Block-3, and /Street-B/Block-4, respectively. Each of these publishers is a street camera that monitors traffic congestion status and periodically generates available content names so that subscribers are aware of the existence of newly created content. If the network resources are sufficient, the subscriber can receive streams from all of the cameras. Otherwise, the subscriber needs to select a subset of the publishers to constitute its data summary. An effective strategy would be to minimize the redundancy of content. In this example, two cameras on the same street are more redundant than others. Thus, one camera of each street can be removed from the summary (unsubscribed) to save some bandwidth.

One extra benefit of having the prefix tree is that subscribers can zoom in or out the topic if necessary. The higher level of nodes in the tree denotes a more comprehensive view of the topic. In contrast, the lower level of nodes represents a more specific topic. For example, if the subscriber specifies '/' as interest, then it gets videos from all cameras, if the network is capable. If the subscriber wants to zoom in, say '/Street-A/Block-1', then it gets videos from only one camera. If the subscriber wants to zoom out again, say '/Street-A,' then it gets streams from two cameras in the street, network conditions permitting. In short, the subscriber drives what range of content is to be summarized.

C. Summarizartion-Based Congestion Control

The idea of summarizing content across several sources motivates a new direction in congestion control, namely, summarization-based congestion control. Traditionally, congestion control mechanisms adaptively control the sending (or retrieving) rate of individual sources according to network conditions. There have been several efforts studying congestion control mechanisms in ICNs. For a representative set of examples, recent work [9] provides a comprehensive survey of the state-of-the-art in NDN congestion handling. Current approaches are categorized into three groups: Receiver-based, Hop-by-Hop interest shaping, and hybrid approaches. Receiver-based algorithms allow the data consumer (terminal) to shape the interest issue rate, whereas hop-by-hop algorithms do so in the network layer. Hybrid approaches combine the two. Pub/Sub-Sum complements prior solutions using an application layer technique.

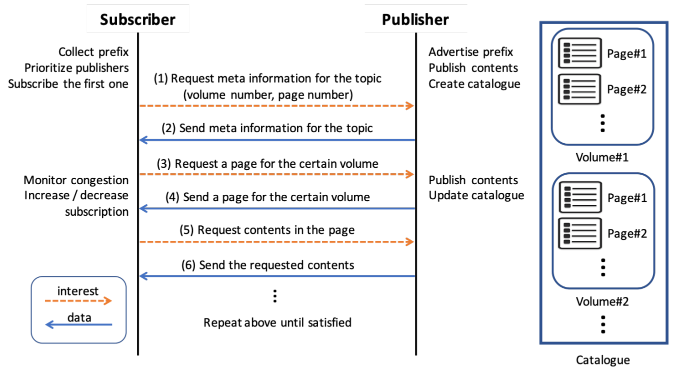
In this work, we combine our application-level congestion control mechanism with one of the receiver-driven approaches, called ICP [10]. This algorithm works similarly to the current TCP congestion control algorithm. It maintains a congestion window that defines the maximum number of interest packets in transit. Initially, it doubles the window size until congestion is detected, which is the slow-start phase. When the congestion is first detected, it halves the window size and switches to the AIMD (additive-increase-multiplicative-decrease) phase. That is, if congestion is not detected, the window size increases gradually. If congestion is detected, the window size decreases quickly. Whenever a consumer issues a new interest packet, it triggers a timer to measure the packet round-trip time (RTT) between the interest issue and the data arrival. If the RTT delay is longer than the expected timeout delay, it regards that congestion occurs in the network.

If the network gets congested, the consumer signals to the subscriber so that it can adjust its data volume to retrieve; otherwise, the subscriber assumes that the network is sufficient to manage the current volume of data requests. Based on this algorithm, we develop an enhanced congestion handling solution as follows. A subscriber interested in data under some tree prefix first starts by retrieving data from only one of the branches in the tree (one branch subscription). If the network has no congestion for a certain period, the subscriber increments the number of branches from which data is requested by picking a branch that is least redundant with the current subscriptions (i.e., that has the shortest shared prefix with them). We say that a new branch subscription is thus added. The retrieved summary is collected from all subscribed branches of the tree defined by the original prefix. The algorithm for incrementing branch subscriptions is merely a tree traversal that always selects the least-visited branch first with a left-most tiebreaker rule. In Figure 2, if /Street-A/Block-1 is subscribed, then the algorithm picks /Street-B/Block-3 next. On the other hand, if network congestion is detected, it decrements subscriptions.

III. Implementation

This section describes implementation details for the design mentioned above. We provide a library package, called *Athena*, that can effectively schedule data retrievals according to the specified application needs [11].

The Pub/Sub-Sum protocol is an Athena layer component placed between the application and transport layers. It offers APIs to publish and subscribe contents under a specific prefix (topic). When an application asks to subscribe to the prefix, the subscriber collects the matching prefixes from the network. Then, it prioritizes the collected prefixes in the least-redundant



Subscriber
Publisher
Collect prefix
Prioritize publishers
Subscribe the first one
(1) Request meta information for the topic (volume number, page number)
Advertise prefix
Publish contents
Create catalogue
(2) Send meta information for the topic
(3) Request a page for the certain volume
Monitor congestion
Increase / decrease subscription
(4) Send a page for the certain volume
Publish contents
Update catalogue
(5) Request contents in the page
(6) Send the requested contents
Repeat above until satisfied
interest
data
Page#1
Page#2
Volume#1
Volume#2
Catalogue

Fig. 3: Control flow of Pub/Sub-Sum protocol

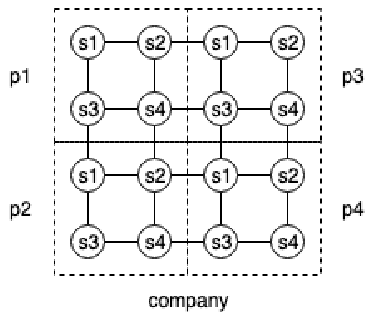


Fig. 4: A grid topology with 16 nodes in Mininet simulation

order based on names and starts subscribing to the very first one in the list.

It first asks the meta information about the served contents set including the volume number and its page number as described in Section II-A. Depending on the application type, it can select either the first volume or the last one depending on application needs. If an application does not care about historical data, it will choose the latter. Once the volume number is decided, it retrieves the first page of the volume to get a prioritized contents list. Finally, the subscriber can request actual contents based on names in the given page. This process is iterative until the application cancels the subscription. The visualization for this process can be found in Figure 3.

Given the prefix of a content tree, the protocol starts with activating one branch subscription, then gradually increases the number of branch subscriptions if no congestion is detected for a certain period. If the subscriber gets congestion signals from the underlying consumer, it starts a timer to measure the round-trip time (RTT) between interest packet departure and data packet arrival. If the RTT is longer than the expected, it re-issues an interest packet up to some threshold. This retransmission algorithm is the same as TCP as described in RFC-6298 [12]. If any time-out occurs, the consumer signals to the subscriber so that it can know that congestion is suspected. Upon receiving this signal, the subscriber decreases the number of active branch subscriptions adaptively until congestion is gone.

IV. Evaluation

This section evaluates the system in two different environments. The first environment is Mininet, a widely used network simulator that enables one to flexibly configure the network environment including link speeds and network size [13]. The second one is a real device testbed using Raspberry Pis. The purpose of the testbed is to show that the protocol can run on actual devices and that empirical results match simulations. Detailed setup and experiment results are described in the following subsections.

A. Mininet

In this experiment, we have composed a hypothetical scenario for better understanding of the use of the Pub/Sub-Sum protocol. The scenario is about company reconnaissance, where a company is searching a particular area, and the company commander wants to have periodic reports from sub-units. For convenient naming, let's say every 4th unit is a headquarter (e.g., 4th platoon is the company HQ). Every squad has a mobile camera and streams images (1MB each) every second for 30 seconds to the company commander. To simulate this, we create the grid topology with 16 nodes as shown in Figure 4. A name for each node is hierarchically composed such as /company/platoon-1/squad-1 for the top-left node. Every squad publishes contents with the sequence number under its prefix. Then, the company commander, located at the bottom-right corner, collects reports from the sub-units by simply specifying '/company' as a topic to subscribe.

Two different algorithms are used in the experiments. The baseline subscribes to 16 publishers all the time while Pub/Sub-Sum adaptively controls the number of subscriptions by considering the network conditions. Two different network speeds are tested: 100 and 10 Mbps. Observe that subscriptions reflect the volume of *requested* (as opposed to *delivered*) content. It is a measure of input load imposed on the network, where a higher load is not necessarily better.

We consider non-redundant content items. In this experiment, content generated by the same platoon at the same time slot is considered redundant. In other words, since the squads in the same platoon are geographically close to each other, their reports can be considered as redundant. Non-redundant items are those generated at different times or by different platoons. In this case, the difference between Pub/Sub-Sum and the baseline becomes more dramatic. Figure 5 shows that the number of collected non-redundant content items (by the above definition) from each platoon over time. The result denotes that Pub/Sub-Sum collects non-redundant reports faster than the baseline in both network environments. Notably, in the constrained network, Pub/Sub-Sum achieves more than 75% information coverage (i.e., collects more than 75% of all non-redundant items) at the end while the baseline attains lower than 50%. This result expresses the main strength of Pub/Sub-Sum; quickly summarizing content by requesting less

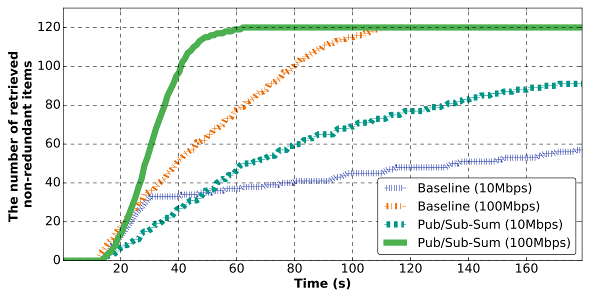


Fig. 5: The number of retrieved non-redundant items over time for the baseline and Pub/Sub-Sum in 10/100 Mbps networks

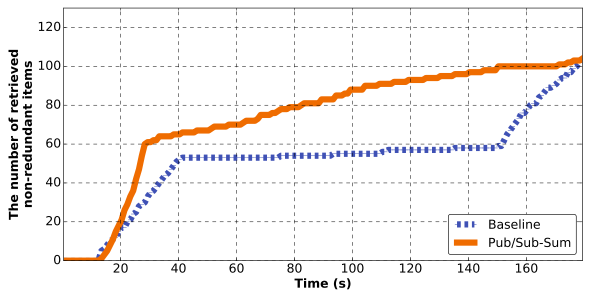


Fig. 7: The number of retrieved non-redundant items over time for the baseline and Pub/Sub-Sum in the congested network

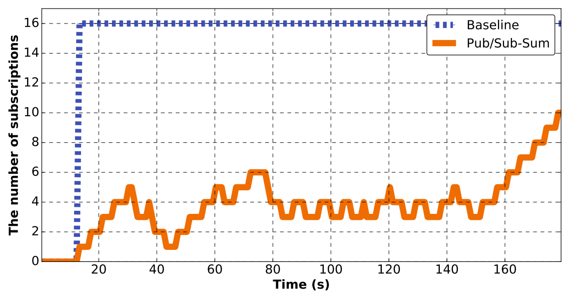


Fig. 6: The number of active subscriptions over time for the baseline and Pub/Sub-Sum in the congested network

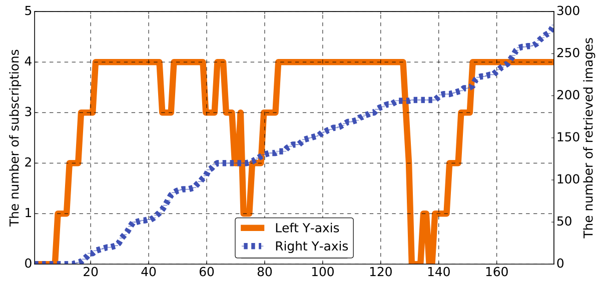


Fig. 8: The number of active subscriptions and retrieved items over time for the baseline and Pub/Sub-Sum in the Raspberry Pi testbed when the network is congested

redundant data objects first.

The next part of the evaluation explores a dynamic network environment, where the network speed starts with 100Mbps but drops after 30 seconds to 2Mbps (e.g., due to an attack such as jamming). The congestion period is 2-minute long followed by 30 seconds of normal operation. We simulate such an environment by controlling link bandwidth through Mininet APIs.

Figure 6 shows how the adaptive algorithm behaves for the denied network. Until the congestion detected, it linearly increases the number of subscriptions. This number drastically drops during the congestion period then stays around 4. After the congestion is gone, the algorithm quickly increases the number of subscriptions because no congestion is found. Figure 7 shows the counts of the non-redundant reports from platoons. As before, Pub/Sub-Sum achieves better information coverage than the baseline especially during the congestion period.

B. Raspberry Pi Testbed

The Mininet simulation shows that the implementation works well as designed. In the next experiment, we want to verify that it operates on real devices as well and to see whether the simulation results match with the real testbed case. To this end, we have constructed a simple testbed with 4 Raspberry Pis as publishers and a desktop as a subscriber. Raspberry Pis equipped with cameras publish a snapshot image every second. Two of them see one side of a scene, and the remaining two see the other side as can be inferred from the node names in the figure. The desktop, which is the subscriber, then collects the published images and display them on the screen whenever being updated.

Using the testbed, we test a congestion scenario similar to the previous experiment. Since the testbed currently supports 100Mbps full duplex for Raspberry PIs and 1Gbps full duplex for the desktop, we need to manually downgrade the speed to simulate congestion. One way to do so is to use *ethtool*, a Linux utility to modify parameters of network interface controllers. Using the tool, we could downgrade the speed to 10Mbps half duplex. Different than the Mininet method, *ethtool* first disconnects the network to reset the rate. That is why there are two valleys between congestion and non-congestion zones in the graph.

The figure explains that the subscriber reaches the maximum network capacity when the network gets congested. However, the congested network is still able to get streams from all four cameras as in Figure 8. When *ethtool* changes the network speed the number of subscriptions drop, but it recovers to 4 quickly when the network is reconnected. The number of retrieved images is higher in the constant phase while the speed gets a bit slower during the congestion period.

V. Related Work

Data summarization to save network resources has been studied from data compression to data sampling. This work broadly falls in the latter category. Especially, [14] propose similarity based image sampling technique in a disruption-tolerant network and [15] extends the work by generalizing the content type.

The most relevant work to this is InfoFunnel, InfoMax [16], [3], [4]. Both protocols are designed to operate on ICNs and exploit naming hierarchy to minimize contents redundancy. InfoFunnel is to collect content from multiple producers while InfoMax is to disseminate contents to multiple consumers. Pub/Sub-Sum provides a similar function while it is designed as a pub-sub service for many-to-many.

Since the idea of ICN was first introduced in [5], several instances were proposed and NDN is considered to be one of the leading architectures [6]. Several work have been proposed to study congestion handling in NDN. According to [9], they can be grouped into three: receiver-driven, hop-by-hop interest shape, and hybrid. Receiver-driven is to have a consumer side detect congestion by RTO timeout and shape interest rate [10]. Hop-by-hop interest shaping let intermediate nodes control the rate based on queue length [17] or incoming interest rate [18]. Hybrid between two are studied in [19]. This work is differentiated than above in that it relieves congestion by reducing redundant contents injected to the network.

VI. Conclusions

This work describes Pub/Sub-Sum, a content sampling-based pub-sub pattern on ICNs. It selectively subscribes to the available publishers in the network by considering source redundancy. This work extends the previous work, InfoMax, in that Pub/Sub-Sum diversifies not only content for a particular source but also sources themselves. It also describes how to handle network congestion by adaptively controlling the number of subscriptions, jointly working with the traditional approach that shapes the interest rate. In the future, we plan to extend this work to satisfy the real-time constraints from applications so that the algorithm optimizes the network resources while meeting the given constraints.

Acknowledgements

Research reported in this paper was sponsored in part by the Army Research Laboratory under Cooperative Agreements W911NF-09-2-0053 and W911NF-17-2-0196, in part by DARPA under award W911NF-17-C-0099, and in part by NSF under grants CNS 16-18627 and CNS 13-20209. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, DARPA, NSF, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] S. Gupta and S. K. Gupta, "Abstractive summarization: An overview of the state of the art," *Expert Systems with Applications*, vol. 121, pp. 49 – 65, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418307735

[2] N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," in *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, Jan 2017, pp. 1–6.

[3] J. Lee, A. Kapoor, M. T. A. Amin, Z. Wang, Z. Zhang, R. Goyal, and T. Abdelzaher, "InfoMax: An information maximizing transport layer protocol for named data networks," in *International Conference on Computer Communication and Networks (ICCCN'15)*. IEEE, August 2015, pp. 1–10.

[4] ——, "InfoMax: A transport layer paradigm for the age of data overload," in *Advances in Computer Communications and Networks - from green, mobile, pervasive networking to big data computing*, J. Fagerberg, D. C. Mowery, and R. R. Nelson, Eds. River Publisher, 2016.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT'09)*. ACM, December 2009, pp. 1–12.

[6] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," in *SIGCOMM Computer Communication Review (CCR)*. ACM, July 2014.

[7] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "Nlsr: Named-data link state routing protocol," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 15–20. [Online]. Available: http://doi.acm.org/10.1145/2491224.2491231

[8] Y. Li, A. Afanasyev, J. Shi, H. Zhang, Z. Zhang, T. Li, E. Lu, B. Zhang, L. Wang, and L. Zhang, "Ndn automatic prefix propagation," Department of Computer Science, UCLA, Tech. Rep., March 2018. [Online]. Available: https://named-data.net/publications/techreports/ndn-0045-1-auto-prefix-propagation/

[9] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking – a survey," *Computer Communications*, vol. 86, pp. 1 – 11, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416301566

[10] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp: Design and evaluation of an interest control protocol for content-centric networking," in *2012 Proceedings IEEE INFOCOM Workshops*, March 2012, pp. 304–309.

[11] J. Lee, K. Marcus, T. Abdelzaher, M. T. A. Amin, A. Bar-Noy, W. Dron, R. Govindan, R. Hobbs, S. Hu, J.-E. Kim, L. Sha, S. Yao, and Y. Zhao, "Athena: Towards decision-centric anticipatory sensor information delivery," *Journal of Sensor and Actuator Networks*, vol. 7, no. 1, 2018. [Online]. Available: http://www.mdpi.com/2224-2708/7/1/5

[12] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing tcp's retransmission timer," Internet Requests for Comments, RFC Editor, RFC 6298, June 2011, http://www.rfc-editor.org/rfc/rfc6298.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6298.txt

[13] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[14] M. Y. S. Uddin, M. T. A. Amin, T. Abdelzaher, A. Iyengar, and R. Govindan, "Photonet+: Outlier-resilient coverage maximization in visual sensing applications," *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, April 2012.

[15] S. Wang, S. Hu, S. Li, H. Liu, M. Uddin, and T. Abdelzaher, "Minerva: Information-centric programming for social sensing," *Proceedings of the 22nd International Conference on Computer Communications and Networks*, pp. 1–9, July 2013.

[16] S. Wang, T. Abdelzaher, S. Gajendran, A. Herga, S. Kulkarni, S. Li, H. Liu, C. Suresh, A. Sreenath, H. Wang, W. Dron, A. Leung, R. Govindan, and J. Hancock, "The information funnel: Exploiting named data for information-maximizing data collection," in *Proc. International Conference on Distributed Computing in Sensor Systems*, May 2014.

[17] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *2012 Proceedings IEEE INFOCOM Workshops*, March 2012, pp. 322–327.

[18] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/2491224.2491233

[19] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 491–496, Sep. 2012. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377772