Filling the Gaps of Unused Capacity through a Fountain Coded Dissemination of Information

George Parisis
Computer Laboratory
University of Cambridge
Cambridge, UK
george.parisis@cl.cam.ac.uk

Dirk Trossen [*]
TecVis LP
Colchester, UK
dot@tecvis.co.uk

ABSTRACT

Lowest Cost Denominator Networking (LCDnet) envisions *"breaking the mould of thinking that law of economics should govern connectivity to all"*. It brings together a multi-layer resource pooling of Internet technologies at several levels to support benevolence in the Internet. One of the proposed levels of resource pooling involves better network and storage utilisation, as promised by Information-centric networking architectures. In this paper we present a transport and resource management approach on top of an information-centric network that enables efficient, multi-source and multi-path information dissemination as well as in-network caching and mobility support, characteristics that are well desired in the LCDnet context.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*distributed networks, network communications*

Keywords

Digital fountains; information-centric networking; multi-path and multi-source information dissemination

1. INTRODUCTION

It is the LCDnet [13] vision to establish access to the Internet as a basic human right. At the technological level, an improved utilisation of temporarily unused communication resources is seen as the key to increase the possibility for affordable Internet access in situations where traditional economic models do not suffice. For this, utilizing distributed computing and storage resources is crucial for filling the gaps of unused capacity.

[*]The research presented in this paper was conducted while working as a Senior Researcher in the Computer Laboratory, University of Cambridge.

In recent years, Information-Centric Networking (ICN) has provided an alternative to the current IP-based Internet architecture with similar goals to LCDnet, i.e., trading off communication, computing and storage for optimal dissemination across distributed end nodes. Core to efforts such as [6, 16, 8, 15] is the focus on information as the main principal of communication. This focus is based on the recognition that WHAT is communicated is often more important than WHO is communicating.

When focusing on the WHAT of a communication, the notion of well specified *endpoints* becomes less important, moving the mental model from a well-defined endpoint-to-endpoint model to a loosely coupled multipoint one, where information can be contributed by many sources for many receivers. Managing the resources of the underlying network becomes a challenge now since the application requirements of general multipoint dissemination scenarios are diverse and, therefore, difficult to address with a single solution [5]. This diversity is reinforced by new application scenarios such as sensor and social networks. So any approach to resource management must cater to this large variety of application requirements, in particular when trying to achieve an affordable access model to these applications from the perspective of the LCDnet vision.

Despite the shift from endpoints to information being delivered across networks, recent efforts on resource management within information centric networks, such as [1, 3], have been heavily inspired by well-known flow based solutions in the IP world. Researchers seem to be stuck with the notion of well-defined flows between well-identified network entities, re-applying the principles of TCP-like mechanisms in an information-centric world.

In this paper we take a different approach by focusing on the information to be disseminated rather than the flows between the entities disseminating this information. For this, we utilize *fountain coding* [9] as an information theoretic approach to encoding content, extending our introductory work in [11]. Unlike work in reliable (IP) multicast transport [10, 2] and work on multipath support using erasure coding [14], we directly embed the fountain codes into the (information centric) identification scheme used at the networking layer. For this, encoded symbols are separately identified in a *self-contained* manner. Contrary to reliable IP multicast where encoded symbols are useless after transmission, in-network caches in our approach can be used to improve performance by replaying encoded symbols, in particular at times when it is more efficient from an overall utilisation perspective (e.g., at night). Subscribers' mobility can be

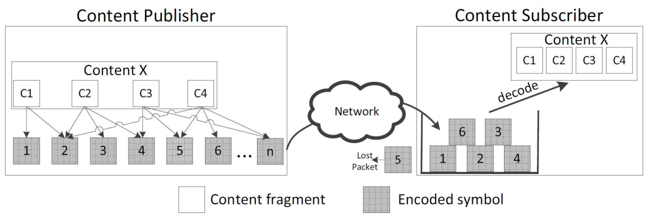

Figure 1: Fountain Coding Information Items

also assisted by pro-actively caching symbols at predicted attachment points.

At the heart of our approach, we utilize the strong separation of core network functions provided by our ICN architecture [15], decoupling the usage of fountain codes and the management of the inherent multipoint relationships for the dissemination of the coded information. This decoupling is a fundamental difference to the mapping onto multicast groups, opening up new opportunities for resource management, such as utilizing cached parts of the information or spreading the information delivery across multiple paths. In this paper, we describe how multiple publishers and network caches can simultaneously publish encoded symbols to subscribers via multiple paths. Subscribers are also able to receive symbols via multiple network interfaces.

In order to present our contribution, we provide an overview of digital fountains in Section 2. In Section 3, we discuss the enabling factors for efficient information dissemination using fountain coding within our ICN architectural context. We then elaborate on how our approach supports multi-publisher and multi-path dissemination as well as caching and mobility (Section 4). In Section 5, we present our prototype implementation and experimental results that illustrate the feasibility of our approach along with its current performance in a Gigabit testbed. We conclude our paper in Section 6.

2. DIGITAL FOUNTAINS

Digital fountains involve coding of information. Receivers are able to decode the initial content once they have received slightly more symbols, than the fragments (*input symbols*) in the original content. As shown in [4], by optimally setting the necessary parameters, the overhead can be as low as 5%. As depicted in Figure 1, a sender stores the content to be encoded and for each encoded symbol calculates how many fragments (of constant and known size) of the content will be encoded in the symbol; this is called the *degree* of the encoded symbol.

For example, the degree of encoded symbol *2* is 3. Selecting the degree is the most crucial step in the process since it affects the efficiency of the information delivery in terms of the number of required symbols to decode the content and the decoding complexity. The sender uniformly selects degree number of fragments and encodes them into the final symbol by XORing them; this set is called the symbol's neighbours. For instance, the neighbour set of symbol *2* includes fragments *C1*, *C2* and *C4*. A receiver utilizes symbols with degree 1 to partially or fully decode other symbols by XORing them with the decoded symbol. In Figure 1, the receiver utilizes the encoded symbol *1*, to decode fragment *C1*. It then XORs it with symbol *2*. *C2* is decoded using symbol *3*, which is also XORed with symbol *2*. Symbol *2* now only contains *C4*, which is decoded.

Lost packets do not affect the decoding process as long as encoded symbols continue to be received and no feedback channel is required. Note that a sender can produce a very large number of encoded symbols and continue sending them as long as interested receivers exist. The only information that must be known by the receiver is the neighbours' set of each symbol.

3. ENABLING EFFICIENT INFORMATION DISSEMINATION

The architectural context underpinning our work is based on [15, 16]. An implementation of an in-kernel network stack which follows our architectural context is presented in [12]. For simplicity, we assume a dissemination strategy, in which information is disseminated across a single network domain. Intra-domain rendezvous can be realized in dedicated network nodes that may share or replicate the information structure and the interested nodes (publishers and subscribers). Topology management is realized in dedicated nodes that know the network topology and are notified when nodes attach to or detach from a forwarding node. Depending on the domain size, one or more cooperating topology managers may be required. Finally, forwarding is realized using LIPSIN [7] identifiers, which natively support source-based multicasting in small to medium scale networks.

We see the following aspects within our architectural context as crucial for addressing the challenges in multipoint, fountain-based delivery. Firstly, the publish/subscribe service model naturally supports the notion of digital fountains for which a subscriber may be interested. This is similar to solutions in the IP multicast space, such as [10]. However, information-centric networking does not single out multicast as an exception but as a norm, unifying the service model across multicast and unicast delivery relations alike.

Secondly, the core functions separate the concerns of tree management from that of delivery of the (coded) information. Hence, while we can utilize efficient stateless multicast solutions, such as [7], the separate topology formation and management function can realize opportunities for multi-source and multi-path dissemination as well as in-network caching. With the (network) resource owner receiving the most benefit for such optimized resource management, realizing these opportunities within the topology management function seems almost natural and is indeed the core of our proposal (see Section 4).

Thirdly, the identification principles of labelling and scoping information [15] allow for embedding algorithmic relationships in the delivery relations even after the initial demand/supply match occurred (through the rendezvous function) and the initial delivery relationship has been established (through the topology management and formation function). With that, LIPSIN identifiers can be reused for publishing algorithmically related information items.

Finally, identifying information items using statistically unique labels allows for caching individual encoded symbols in network nodes, which in turn can play them out again. Embedding the information for decoding the symbol into the identifier (through an algorithmic relation) makes these in-network stored symbols self-contained.

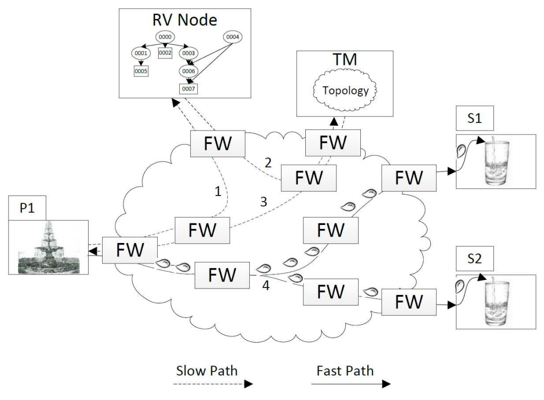

Figure 2: Intra-Domain Fountain (Single Fountain)

4. DIGITAL FOUNTAINS IN ICN

Let us now outline the usage of digital fountains in our architectural context. We first outline the basic operation before detailing the multi-source/path and caching support.

4.1 Basic Operation

Let us first apply the fountain coding principles within our architectural context in order to provide reliable and efficient transport among communicating entities. We base our description on *LT codes* [9], although other rateless coding schemes could be used as well.

We utilize the principles of information labelling and scoping to create labels for all encoded symbols that can be produced when encoding some content. These labels are used to communicate encoding-related information that is necessary for receivers (subscribers) to decode each symbol by calculating its degree and neighbours. We call such labels *algorithmic*, because they are all produced according to an algorithm that must be known by the communicating entities. Subscribers utilize the same algorithm to extract meaningful information out of the identifier, which will be later used to decode the initial content.

Statistical uniqueness of labels is very important because they can be cached by any intermediate node and replayed even if the initial content is not decoded. Such cached publication is self-contained because it contains an encoded symbol as well as enough information in the identifier to calculate the degree and neighbours of the symbol.

For our basic operations, we assume, for reasons of simplicity, a single network domain, running a centralized topology manager and a dedicated rendezvous node that manages the information space (Figure 2). To keep the figure readable, we further assume that subscribers *S1* and *S2* (shown with the glass waiting for drops from the fountain) are already subscribed to an item with ID */A/B* (or its parent scope */A*). Publisher *P1* (illustrated as a fountain), then advertises the information item */A/B* (step 1). The rendezvous node receives the request, matches the publisher with the subscribers and publishes a topology formation request to the domain's topology manager (step 2). The request is received by the topology manager, which is subscribed to a scope under which this information item resides, and knows the domain's topology. It creates a LIPSIN identifier from the publisher to the subscribers and publishes it to the publisher (step 3). Finally, the publisher application is notified that subscribers exist.

At that point, the publisher's network stack stores a mapping of the information item to the LIPSIN identifier. The digital fountain is subsequently enabled. Steps 1 to 3 comprise the slow-path operation of our ICN architecture. Step 4 takes place in the fast-path of our network. The publisher utilizes the previously created LIPSIN identifier to publish encoded symbols to the subscribers. Hence, there is no need for rendezvous for each encoded symbol, removing any extra delay. This identifier may be internally updated (in the network stack) as subscribers join or leave the digital fountain (steps 2 and 3 are repeated). This is transparent to the publishing application.

The publisher starts creating and publishing encoded symbols using algorithmically created identifiers (depicted as drops in Figure 2). The only coding-related information that needs to be communicated between the publisher and subscribers is the seed that the publisher uses to encode a symbol. The publisher (pseudo) randomly generates such a seed and, then, using this seed, it calculates the degree and neighbours of each encoded symbol according to the distributions defined in [9]. The publisher must also encode in the information identifier whether the publication contains the last fragment of the content or not (one bit in the label suffices). Finally, it calculates the encoded payload by XORing all neighbours and publishes the (nth) symbol using the identifier $/A/B/algID_n$.

The subscriber's network stack stores a subscription for the item with identifier */A/B* (or its parent scope */A*) and pushes all encoded publications with identifiers $/A/B/algID_n$ to the interested application. The subscriber extracts the seed from the label and calculates the degree and neighbours of the received symbol, assuming that the same pseudo-random number generator is used. If the published item is flagged as containing the last fragment of the initial content, the subscriber also learns the actual size of the transported content when decoding this symbol. Decoding takes place as described in [9]. Fountain codes can produce a very large number of encoded symbols. A subscriber can start receiving symbols at any point and can un-subscribe when it receives the last required symbol for decoding.

4.2 Multi-source and Multi-path Support

According to Section 4.1, a single digital fountain publishes encoded symbols as long as subscribers exist. The topology management and formation function creates a single multicast tree from the publisher to the subscribers, over which all encoded symbols are transferred. Given this approach of separating the resource management from the resource usage (i.e., the actual forwarding), we can envision the creation of multi-source and multi-path relationships, both of which utilise recognised unused capacity in the regions where dedicated sources are added to the delivery or where dedicated paths are utilised (e.g., paths that are otherwise under-utilised). Note that we leave out the details of how such underutilisation can be detected.

However, nothing prevents the support for multiple publishers sending encoded symbols to multiple subscribers over multiple paths and network interfaces. The clean separation of our core network functions allows for transparently creating multiple paths from a publisher to a set of subscribers by just altering the topology management and formation func-

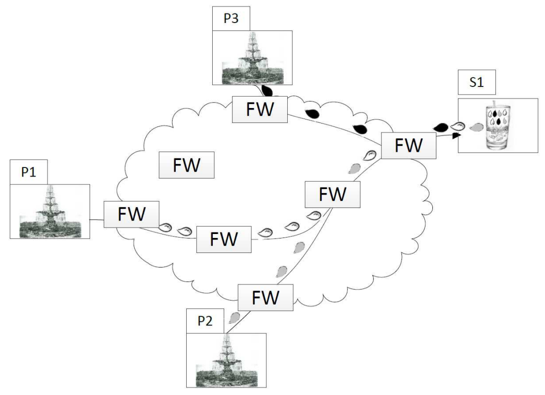

Figure 3: Multi-Source Information Dissemination

tionality. Rendezvous and forwarding functions do not require any change for this. Figure 3 illustrates a multi-source scenario, where a number of publishers store a decoded version of some content.

Subscriber *S1* declares its interest to that content, rendezvous takes place and the topology manager creates three separate paths, one for each publisher, along with the respective LIPSIN identifiers. All publishers are notified and start publishing encoded symbols of the initial content (shown as white, grey and black drops in Figure 3), as described in the previous section. There is a single requirement to support this communication scenario in an efficient way: publishers must not create duplicate encoded symbols (i.e. with identical information identifiers) among each other; that is, their pseudo-random generators for the seed creation (see Section 4.1) must not overlap with each other. This is not difficult to achieve since generators like the *Mersenne-Twister* have very large periods and some more randomness can be always added. This way all encoded symbols received by S1 will be statistically unique. The fountain coding principles dictate that the subscriber will be able to decode the initial content as long as it receives the required number of symbols, irrespectively of the actual source of each symbol.

Multi-path transport is also feasible with the support of the topology management and formation network function. As shown in Figure 4, publisher *P1* publishes symbols to both subscribers S1 and S2 using two different multicast trees. Subscribers can potentially receive encoded symbols from different network interfaces, as also illustrated in Figure 4. In order to do so, a publisher advertises two different information items to the rendezvous function.

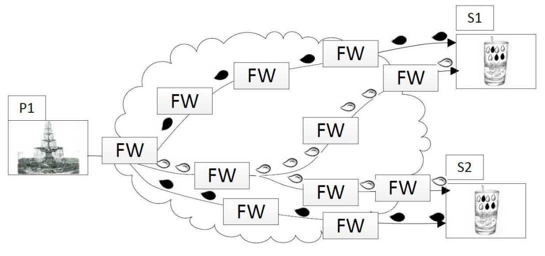

Figure 4: Multi-Path Information Dissemination

Following the example presented in Section 4.1, the publisher advertises items $/A/B$ and $/A/MultiPathAlg_B$, where $MultiPathAlg_B$ is an algorithmic identifier produced by an algorithm known by the topology management and formation function. Subscribers *S1* and *S2* also know this algorithm and separately subscribe to both advertised information items (or the parent scope); therefore, rendezvous takes place twice. However, since the topology management function knows about this algorithm, it can infer that both information identifiers refer to the same content and create two different multicast trees from the publisher to the subscribers. *P1* publishes symbols for both advertised items with identifiers $/A/B/FountainAlgID_n$ (white drops) and $/A/MultiPathAlg_B/FountainAlgID_n$ (black drops). Since subscribers know both algorithms, they can combine all received symbols to decode the initial content with identifier $/A/B$.

4.3 Caching and Mobility

Caching is another method to increase the utilisation of otherwise under-utilised parts of the network. For this, cached symbols are played out by intermediary nodes instead of relying on the original source. Firstly, the topology management function can proactively insert caches as subscribers in the delivery relationship, therefore filling up the caches for future delivery in anticipation of a possible under-utilisation of nearby resources. Secondly, the topology manager can, in reaction to actual detected under-utilisation within parts of the network, insert caches as publishers in the deliver relationship, better utilising the resources in the proximity of the cache.

In-network caching can be achieved by the topology manager creating LIPSIN identifiers from a publisher to one or more subscribers that also include special link identifiers that "connect" the forwarding fabric of a network node up to the network stack. Since all information items are identified using statistically unique labels, it is easy to control which fragments to cache, while the separation of functions provides the control of where the appropriate caching resource should be used (e.g. by explicitly adding one or more paths to caching points into the LIPSIN identifier).

There are several approaches for caching and replaying encoded symbols. In the simplest case, an in-network node decodes the entire content before advertising it. After that, it becomes a regular publisher, which can be later considered when a forwarding path is formed by the topology manager. Moreover, a replaying node can advertise a symbol without first decoding it. This means that it can only replay the encoded symbols that it currently stores. Note that these publications may not suffice to decode the initial content and, therefore, special care must be taken when forming multicast trees for which such a replaying node is the root. All fountain coding schemes provide some statistical guarantees about the number of required symbols. Hence, a replay node might advertise cached information items only after the statistical requirements are met.

Finally, another approach exists for the multi-publisher scheme as described in Section 4.2. In this case, the topology manager could notify multiple publishers to start publishing encoded symbols. Some of these publishers could be in-network nodes that store only encoded symbols. Such a replay node could publish all its encoded symbols and stop when all subscribers have received the data or when it runs out of encoded symbols. Publishers that hold the initial content will continue producing encoded symbols until all subscribers receive the content and un-subscribe.

In [17] the authors show how mobility can be assisted by on and off path caching in our architecture. Our fountain coding mechanism fits very well in that context. Encoded symbols can be cached (or pre-fetched) at predicted attachment points of a mobile ICN node. A node can then start receiving encoded symbols upon its attachment to a new access point, while the topology management and formation function re-calculates a path from one or more publishers to the mobile node. Digital fountains ensure that as long as these symbols have statistically unique identifiers and, therefore, contain statistically unique encoded information, the mobile node can utilize received symbols from multiple attachment points to decode and reconstruct the initial content.

5. EXPERIMENTAL EVALUATION

We now present an experimental evaluation of our prototype implementation of our approach. The mechanisms described in this paper are implemented as a C++ library that utilizes the publish/subscribe service model that is exported by *Blackadder*, our ICN network stack implementation [12]. We have used the LT codes in our prototype and implemented the robust soliton distribution, as described in [9].

It is our intention in this section to show the feasibility of our approach as well as its current performance characteristics and encoding overhead in a high-speed network environment. For this, we have utilized a Gigabit Ethernet testbed, on top of which we have a deployed a simple ICN star topology. In the middle node the Rendezvous and Topology Management and Formation functions are implemented. All satellite nodes act as publishers or subscribers, depending on the experiment.

In Figure 5, we present the application throughput and goodput (in MB/sec) when a single publisher publishes encoded symbols for a previously advertised information item. The application goodput is the rate at which the subscriber receives useful data; i.e. the size of the content divided by the time to fully decode it. The application throughput is the rate at which the subscriber receives and decodes input symbols; i.e. the total size of the received symbols divided by the time to fully decode the content.

Although our approach allows for a subscriber to join at any time, we synchronize the subscribers in our experiment by having them subscribe to the item before the publisher advertised it. The content consists of 10000 fragments (input symbols) of 1400 bytes each. For parameters *c* and *Ît'* of the robust soliton distribution [9], we have utilized the values presented in [4], providing close-to-optimal behaviour. All experiments have been repeated multiple times and the average values are presented. Note that in all subsequent figures, the error bars represent the 95% confidence interval for our measurements.

In Figure 5, we observe that when a single subscriber exists, the average application goodput is 10.5 MB/sec. As the number of subscribers increases, the total application goodput is linearly increased to a value of 65 MB/sec since our ICN architecture natively supports multicasting based on LIPSIN identifiers and Blackadder can forward and process publications (here encoded symbols) at Gigabit speeds [12]. Regardless of the number of simultaneous subscriptions, and

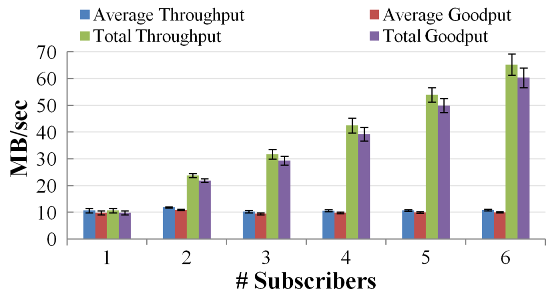

Figure 5: Performance: Publisher - Multi-Subscriber

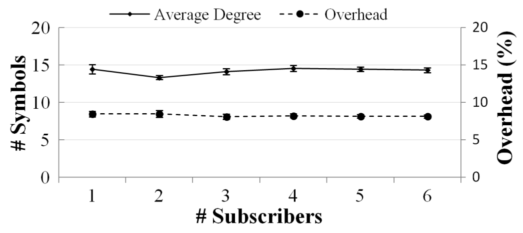

Figure 6: Coding Overhead: Publisher - Multi-Subscriber

as we clearly show below, the network overhead, indicated by the difference between the application throughput and goodput, is constant.

Figure 6 presents the average degree of all encoded symbols published in the previous experiment as well as the network overhead imposed by sending more encoded symbols than the actual number of input symbols (i.e. fragments of the initial content). Both the average degree and the network overhead do not depend on the number of subscribers that simultaneously receive the encoded content. In fact, both parameters are only dependent on the distribution used for calculating the degree of each symbol, along with its specific parameters. Here we observe that the overhead is 8%, slightly larger than the one reported in [4]. The average degree is 13 to 15 symbols. Note that the median degree was always 2. This means that most of the time only 2 input symbols were XORed to produce an encoded symbol, except for a few times when a large number of input symbols were chosen (i.e. exactly as the robust soliton distribution suggests).

The bottleneck in our prototype is the decoding process, which can reach a data rate of 11.5 MB/sec (application throughput), as observed in Figure 5. The encoding process is much faster. We observed that a single publisher is able to publish encoded symbols at 35MB/sec. However, there exists vast space for optimisation. Although Blackadder runs in kernel space, the fountain coding library runs in a single thread, without parallelising the XORing of symbols. However, both the encoding and decoding is ideal for hardware offloading. Encoding a symbol can take place in parallel by XORing multiple input symbols in different cores and gradually combining the results to a final symbol. The decoding process can be also pipelined, since upon reception of a symbol with degree 1, all other, yet un-decoded, symbols must be processed one by one.

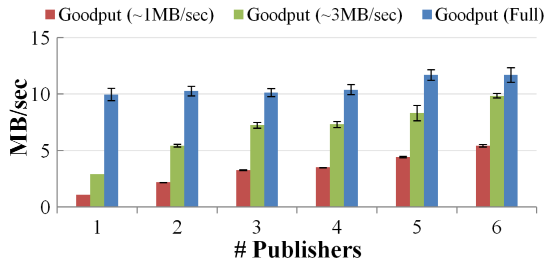

Figure 7: Performance: Multi-Publisher - Subscriber

Figure 7 shows the results when measuring the average goodput of a single subscriber when multiple publishers publish encoded symbols of the content. Here, the topology manager selects up to 6 publishers. Publishers randomize their selection of seeds for calculating the degree and neighbours' set of each symbol (and their algorithmic identifiers). The blue bars represent the average goodput when all publishers publish encoded symbols at full rate. When more than 3 publishers are notified, the Gigabit link is saturated. However, the average goodput, limited by the decoding process, was measured at 11 MB/sec for all different number of publishers. In order to show the performance increase in the multi-publisher scenario, we conducted the experiment by throttling the encoding rate at 1 MB/sec (red bars) and 3 MB/sec (green bars). For these cases, we observe that the subscribers utilize encoded symbols sent by all publishers to decode the initial content, thus the measured application goodput increases as the number of publishers and the encoding rate increase, up to the measured decoding rate of 10 MB/sec.

6. CONCLUSIONS

The LCDnet vision of re-using often unused network capacity for making Internet access affordable for those who are currently not connected aligns well with the ambition of ICN to provide a trade-off between communication, computation and storage. We built on this ambition in our paper, proposing to move from a traditional flow model towards a diffusion model that utilizes a fountain coding based identification approach. This makes packets sent in our network self-contained, allowing for caching, multi-path/multi-subscriber support, ultimately utilizing available communication resources whenever they are available to the participating network nodes.

Our prototype-based experiments show the feasibility of our approach with a very promising performance in single- and multi-publisher scenarios. In our future work, we will focus on the algorithms to react to under-utilisation by appropriately steering the diffusion of information through cache-assisted, multi-source/path delivery, provided by the basic operation of our solution in this paper. This work will include congestion control and avoidance mechanism similar to the Asynchronous Layer Coding mechanisms [10].

7. REFERENCES

[1] S. Arianfar, J. Ott, L. Eggert, P. Nikander, and W. Wong. A Transport Protocol for Content-Centric Networks. 2010.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of SIGCOMM '98*, pages 56–67, 1998.

[3] G. Carofiglio, M. Gallo, and L. Muscariello. ICP: Design and evaluation of an interest control protocol for content-centric networking. In *Proceedings of INFOCOM '12 Workshops*, pages 304–309, 2012.

[4] P. Cataldi, M. P. Shatarski, M. Grangetto, and E. Magli. Implementation and performance evaluation of lt and raptor codes for multimedia applications. In *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia*, IIH-MSP '06, pages 263–266, 2006.

[5] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, and M. Luby. The reliable multicast design space for bulk data transfer. *RFC 2887*, 2000.

[6] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. *Commun. ACM*, 55(1):117–124, Jan. 2012.

[7] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: line speed publish/subscribe inter-networking. In *Proceedings of SIGCOMM '09*, SIGCOMM '09, pages 195–206, 2009.

[8] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of SIGCOMM '07*, pages 181–192, 2007.

[9] M. Luby. LT codes. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 271–, 2002.

[10] M. Luby, M. Watson, and L. Vicisano. Asynchronous layered coding (ALC) protocol instantiation. *RFC 5775*, 2010.

[11] G. Parisis and D. Trossen. Digital fountains in information-centric networking. In *Proceedings of SIGCOMM ICN workshop '13*, 2013.

[12] G. Parisis, D. Trossen, and D. Syrivelis. Implementation and evaluation of an information-centric network. In *Proceedings of IFIP Networking '13*, 2013.

[13] A. Sathiaseelan and J. Crowcroft. LCD-Net: lowest cost denominator networking. *SIGCOMM Comput. Commun. Rev.*, 43(2):52–57, Apr. 2013.

[14] V. Sharma, S. Kalyanaraman, K. Kar, K. K. Ramakrishnan, and V. Subramanian. MPLOT: A transport protocol exploiting multipath diversity using erasure codes. In *Proceedings of INFOCOM '08*, pages 121–125, 2008.

[15] D. Trossen and G. Parisis. Designing and realizing an information-centric internet. *Communications Magazine, IEEE*, 50(7):60–67, July 2012.

[16] D. Trossen, M. Sarela, and K. Sollins. Arguments for an information-centric internetworking architecture. *SIGCOMM Comput. Commun. Rev.*, 40(2):26–33, Apr. 2010.

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