SAID: A Control Protocol for Scalable and Adaptive Information Dissemination in ICN

Jiachen Chen*†, Mayutan Arumaithurai†, Xiaoming Fu†, and K. K. Ramakrishnan‡
*WINLAB, Rutgers University, NJ, U.S.A. jiachen@winlab.rutgers.edu
†Institute of Computer Science, University of Göttingen, Germany. {arumaithurai,fu}@cs.uni-goettingen.de
‡University of California, Riverside, CA, U.S.A. kk@cs.ucr.edu

ABSTRACT

Information dissemination applications (video, news, social media, *etc.*) with large number of receivers need to be efficient but also have limited loss tolerance. The Information-Centric Networks (ICN) paradigm offers an alternative approach for reliably delivering data by naming content and exploiting data available at any intermediate point (*e.g.*, caches). However, receivers are often heterogeneous, with widely varying receive rates. When using existing ICN congestion control mechanisms with in-sequence delivery, a particularly thorny problem of receivers going *out-of-synch* results in inefficiency and unfairness with heterogeneous receivers. We argue that separating reliability from congestion control leads to more scalable, efficient and fair data dissemination, and propose SAID, a control protocol for Scalable and Adaptive Information Dissemination in ICN. To maximize the amount of data transmitted at the first attempt, receivers request *any next packet* (ANP) of a flow instead of next-in-sequence packet, independent of the provider's transmit rate. This allows providers to transmit at an application-efficient rate, without being limited by the slower receivers. SAID ensures reliable delivery to all receivers eventually, by cooperative repair, while preserving privacy without unduly trusting other receivers.

CCS Concepts

•**Networks** → **Network protocol design; Transport protocols;**

Keywords

ICN; Congestion Control; Reliability; Out-of-synch

1. INTRODUCTION

Large scale information dissemination applications like video streaming (YouTube, Netflix, *etc.*), online social networks (Facebook, Twitter, *etc.*) and news/entertainment (CNN, BBC, RSS feeds, *etc.*) have become common. Many of these applications have limited loss tolerance and depend on the network to provide efficient, fair and reliable content distribution. While IP multicast was designed for large-scale information dissemination, the inability to have an effective congestion control solution, especially in the presence of heterogeneous capacity to receivers, has been a limitation.

A key goal for publishers sending data at an application-efficient rate across the entire receiver population is to be not limited by the slower receivers. Due to the absence of a network layer mechanism to control the delivery rate at the receiver end, previous solutions have either sought to push all the data onto the path, overlooking congestion and unfairness and using end-end unicast recovery, or seek to slow down the sending rate to the slowest receiver [1]. Alternatively, the use of unicast, with the associated inefficiencies, has become the norm. The use of multicast at the application layer (*e.g.*, SCRIBE [2]) exploits TCP's congestion control mechanisms for ensuring loss-free delivery on an end-end basis. However, such solutions are network topology unaware and achieve lower efficiency (caused by the end-hosts replicating the packets rather than the routers) than what an effective multicast solution could be expected to achieve.

The advent of Information-Centric Networks (ICN) offers us an opportunity to take a fresh look at the potential solution approaches. Content-Centric Networks (CCN [3]) or Named Data Networking (NDN [4]) is a representative ICN approach. Although NDN does not mandate a congestion control mechanism, most of the proposed solutions [5–7] choose to use a TCP-like receiver-driven mechanism to limit the number of requests (window) outstanding from a receiver. Since the network maintains flow balance, where one Interest retrieves at most one Data packet, these mechanisms adapt the window using the Additive Increase Multiplicative Decrease (AIMD) principle, much like TCP [8]. Compared to a sender-driven rate control approach, such a receiver-driven approach has the benefit of the consumer[1] being able to control the receive rate. When considering efficient large scale data dissemination where every piece of data is consumed by a large number of receivers, TCP-like mechanisms for receiver-driven multicast can have significant shortcomings, especially with the path to the receivers having different capacities. With such heterogeneity, a problem we observe is that of receivers being **out-of-synch** *even with optimal policies for managing the cache.* The out-of-synch problem can be briefly described as follows: NDN routers cache contents as they are forwarded. When there is temporal locality of requests from receivers, a router that has the cached content can respond. However, with receiver

ICN'16, September 26 - 28, 2016, Kyoto, Japan

ISBN 978-1-4503-4467-8/16/09…$15.00

DOI: http://dx.doi.org/10.1145/2984356.2984370

[1]Consumer/receiver, and provider/sender are used interchangeably in this paper.

heterogeneity, the requests from receivers even for the same data items can diverge over time. Requests from the faster receivers can be well ahead of those from slower receivers. Eventually, when the gap between the faster and slower receivers becomes too large, their requests can no longer be aggregated at the intermediate routers or satisfied by the cache. The slower receivers' requests will have to be satisfied by the content provider (via retransmissions) as a separate flow. These retransmissions will compete for the bottleneck bandwidth and the overall throughput can therefore dramatically reduce. This is a fundamental issue as long as there are heterogeneous receivers and routers with limited cache sizes. The problem can be exacerbated with scale, thus occurring even more often in the core of the network.

In this paper, we propose a control protocol for Scalable and Adaptive Information Dissemination (SAID) in ICN, a novel mechanism enabling large scale efficient data dissemination. We leverage the receiver-driven framework of NDN with enhancements to overcome the out-of-synch problem. SAID achieves efficiency by separating reliability from congestion control. While our design is framed in the context of NDN, we believe SAID can also be used by other multicast solutions. The contributions of this paper include:

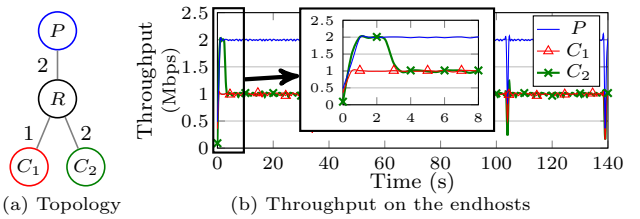
- An analytical and emulation-based study on the out-of-synch problem that shows it will occur in real networks even with large in-network caches as long as there is receiver heterogeneity (see §2);
- A new reliable multicast framework that seeks to maximize the useful throughput, by consumers requesting for *Any Next Packet* (see §3.1). Reliability is then achieved via a repair mechanism that leverages NDN's capability for receivers to request contents from any network node, while preserving privacy and data integrity (see §3.2);
- A receiver-driven congestion control mechanism tailored for ANP delivery that enables *each* receiver to obtain its fair[2] share of the bottleneck link while maintaining an application-efficient sending rate (see §4);
- Evaluations on our prototype and large scale simulations illustrate the benefit of SAID compared to solutions such as ICP and pgmcc (see §5).

2. PROBLEM WITH EXISTING CONGESTION CONTROL – OUT-OF-SYNCH

We first study the out-of-synch problem in NDN and show how it reduces the benefit of in-network caches and the use of pending interests. We demonstrate this with an emulation using CCNx 0.8.0 along with a congestion control mechanism similar to ICP [5]. We show through an analytical model that our observations on the out-of-synch problem is systemic with heterogeneous receivers, and should be expected even with other receiver-driven, in-sequence congestion control mechanisms.

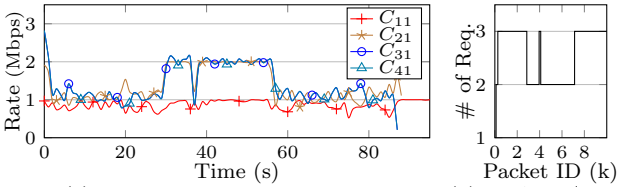
2.1 Demonstration of Out-of-Synch in An Emulated Scenario

To clearly demonstrate the out-of-synch problem and its cause, we use a simple emulation performed in Mini-CCNx. The network topology and the link rates (in $Mbps$) are shown in Fig. 1a. The latency on all the links is $2ms$. Router R has a 50 packets cache[3]. Consumers C_1 and C_2 start to request the same content ($\sim 35MB$ in size, $8,965pkts$) from provider P at the same time. The throughput of the end hosts are shown in Fig. 1b. For the first 2.5 seconds, the PIT and the network cache benefit both receivers. Requests from C_1 either get aggregated or get a cache hit at R. Overall network throughput in this period is $3Mbps$ (sum of downstream link capacities of R). Subsequently, with heterogeneous receiver rates, the receivers' requests deviate farther apart. The requests from C_1 can no longer be satisfied by the cache and they are forwarded to P, to be treated as a distinct flow. The response to these requests start to compete for the bandwidth on the link from P to R and the receive rate of C_2 is thus affected. Since the congestion control protocol tries to achieve fairness between the receivers (flows), the receive rate of the two consumers becomes $1Mbps$ each, and the overall network throughput reduces to $2Mbps$, thus underutilizing the bandwidth by 33%. For the entire transfer, we observed that $< 2\%$ of the requests from C_1 see a cache hit.



(a) Topology (b) Throughput on the endhosts

Fig. 1: Out-of-synch problem emulated in a simple topology.



(a) Receive Rate on consumers (b) # of req./pkt

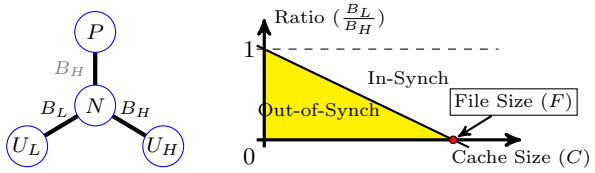
Fig. 2: Out-of-synch: On larger dissemination tree.

Similar results occur in more complex topologies (*e.g.*, Fig. 10 without the dotted links). The consumers C_{11} – C_{41} start to request the same content from provider P_1 at the same time (C_{51} is not active in this emulation). The cache size is $100pkts$ and the content size is $10,000$ packets. The receive rates at the consumers are shown in Fig. 2a and the # of requests (transmissions) per packet observed by P_1 is shown in Fig. 2b. We can see that, similar to the previous simpler case, the consumers once again get out-of-synch soon after the transmission sequence starts. The # of transmissions increase and the receive rate drops. For the first 30 seconds, the aggregate throughput is only around $4Mbps$ (the ideal throughput is $9Mbps$). During the intervals 30–36 and 40–55 seconds, the faster receivers get in-synch again, due to the randomness in the network and cache occupancy. The packet sequences between $2,800$–$3,950$ and $4,100$–$7,100$ are transmitted twice and the throughput of C_{21} through C_{41} increases during these two time periods. But even during this time, the aggregate throughput can only reach $7Mbps$ rather than the $9Mbps$ achievable in the ideal case.

Through these emulations, we see that due to the heterogeneity of the receivers, the cache in the intermediate routers might not be enough to absorb the difference in the request rates of the fastest and slowest receivers. This is the occurrence of 'out-of-synch' problem. When the slower receivers

[2]In this paper, we define fairness as max-min fairness based on the link capacity, which is not affected by the content popularity.

[3]We use a relatively small cache size to quickly demonstrate the out-of-synch problem. However, this is fundamental and occurs even with much bigger caches. Please see §2.2 for the relationship between the cache size and the heterogeneity allowed among the receivers.



(a) Single-branching model (b) Relationship between out-of-synch and request rate, cache size and flow size

Fig. 3: Out-of-synch in a single-branching model.

re-issue requests, these requests are seen as a different 'new' flow, since they can no longer be aggregated or be satisfied from the cache at the routers. These 'new' flows will then compete on the network links with packets of the original flow. In some cases, this would even be with faster receivers on the common links, and affect their download rate as well.

The out-of-synch problem can happen even when all the receivers start their requests for the sequence of data packets at the same time, even with the optimal cache replacement policy. Note that the provider has to re-transmit packets as long as the intermediate router drops the packets within the gap (the difference in the sequence number of the packet requested by the fastest and slowest receiver). When the gap is larger than the available cache size for the flow, no matter which packet the replacement policy chooses, an additional transmission from the provider is required.

2.2 Analysis on Out-of-Synch Occurrence

Receiver-driven feedback-based in-sequence congestion control protocols (*e.g.*, TCP) share the following features: *1*) each data consumer has a local view of the request as if he is the only consumer in the network, *2*) all the data consumers tend to get a (statistically) fair-share of bandwidth, and *3*) the packets in a data object are requested in-sequence and out-of-order is seen as an indication of congestion. We realize that almost all the existing congestion control protocol proposed for NDN fall into this category.

To show the universality of the problem, we generalize the model for congestion control by assuming a best-case scenario where each receiver is receiving a flow of data with a constant bit rate which is exactly the fair-share that receiver can get. We analyze the maximum heterogeneity that can be supported given a certain cache and flow size while the receivers still remain in-synch till the end of the flow. Since we are focusing on a single flow, we also assume a simpler case that the network status (*i.e.*, available bandwidth, cache size and latency) does not change during the lifetime of the flow.

We start with a more precise definition of out-of-synch.

Definition 2.1 (Out-of-synch)**.** *Consider a network with multiple routers interconnected in a tree topology and a flow f that has a provider on the top and receivers at the leaves of the tree. At a branching router N, where available bandwidth to the downstream receivers is in the range $[B_L, B_H]$, the out-of-synch occurs when the difference in the amount of bytes received (the gap, G) between the fastest and slowest receiver in the sub-tree below N is larger than the available cache size C for flow f.*

We then use a single-branch model to demonstrate the relationship among the cache size, flow size and receiver heterogeneity in a simple scenario.

Lemma 2.1. *For a branching router N with cache size C in a dissemination tree, with the request rates of the immediate downstream links in range $[B_L, B_H]$, to avoid out-of-synch, the following condition should hold,* i.e.*:*

$$B_L/B_H \geq 1 - C/F, \tag{1}$$

where F is the size of the flow.

Proof. According to Def. 2.1, to avoid out-of-synch at N, it is sufficient to consider 2 immediate downstream links with the largest and smallest available bandwidth. That is, we can consider the single-branching topology in Fig. 3a such that the two data consumers (U_L and U_H) are requesting for a same flow with size F and their available bandwidth are B_L and B_H ($B_L \leq B_H$).

When the receivers are in-synch, the request sent upstream by N targets a downstream rate of B_H, matching the receive rate of the faster receiver (this is true for all protocols where the network node does not perform an explicit congestion control function and depends on the receivers to generate an appropriate request rate). The download period for U_H is:

$$t = F/B_H.$$

The maximum gap G between the U_L and U_H is therefore:

$$G = (B_H - B_L) \times t = (1 - B_L/B_H) \times F$$

According to Def. 2.1, to keep the consumers in-synch, we need:

$$G = (1 - B_L/B_H) \times F \leq C. \tag{2}$$

Equation (1) is equivalent to (2). □

The requirement for clients in-synch (equation 1) is presented in Fig. 3b. We can see that the requirement for being in-synch cannot be satisfied when the heterogeneity (B_H/B_L) is larger, the flow size F is larger, and/or available cache size C is smaller.

Since the request and data paths in large scale data dissemination form a tree structure rooted at the data provider, we study the in-synch requirements in a k level tree.

Theorem 2.1. *For a dissemination tree with k levels and every intermediate router having cache size C, all the receivers will be in-synch only when the available bandwidth between the fastest receiver and the slowest receiver follow:*

$$B_L/B_H \geq (1 - C/F)^k. \tag{3}$$

Proof. We prove this theorem by contradiction. Suppose that the request rates of the highest and the lowest receivers satisfy

$$B_L/B_H < (1 - C/F)^k, \tag{4}$$

and all the receivers are in-synch.

Without loss of generality, we assume that the receiver with the lowest rate B_L is a downstream consumer of a router N_t at level $t \in [1, k]$. Let $B_{H,t}$ be the highest request rate among the downstream consumers of the router N_t. Note that the consumer with request rate B_H does not have to be the immediate next hop of N_t, the intermediate routers will always forward requests according to the fastest receiver. According to Lemma 2.1, we have

$$B_L/B_{H,t} \geq 1 - C/F. \tag{5}$$

According to (4) and (5), it follows that

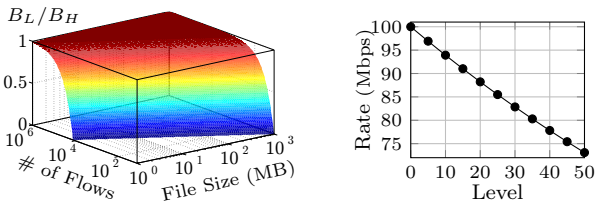
$$B_{H,t} < B_H \times (1 - C/F)^{k-1}. \tag{6}$$

The router N_t is a downstream consumer of a router N_{t-1} at level $t-1$. Similarly, let $B_{L,t-1}$, $B_{H,t-1}$ be the lowest and highest rates among the downstream consumers of the router N_{t-1} respectively. Since $B_{H,t} \geq B_{H,t-1} \geq B_{L,t-1}$, according to Lemma 2.1, we have

$$B_{H,t}/B_{H,t-1} \geq B_{L,t-1}/B_{H,t-1} \geq 1 - C/F. \tag{7}$$

According to (6) and (7), it follows that

$$B_{H,t-1} < B_H \times (1 - C/F)^{k-2}.$$



(a) Relationship among out-of-synch criteria in real router (b) Minimum allowed download rate *vs.* hierarchical level

Fig. 4: Requirements to keep consumers in-synch.

By the similar argument, we can show that

$$B_{H,t-2} < B_H \times (1 - C/F)^{k-3},$$

$$B_{H,t-3} < B_H \times (1 - C/F)^{k-4},$$

$$\ldots$$

$$B_{H,1} < B_H \times (1 - C/F)^{k-t} \leq B_H.$$

Since the highest rate of downstream consumers of the router at level 1 should be B_H, *i.e.*, $B_{H,1}$=B_H, we reach a contradiction and the proof is completed. □

From Theorem 2.1 we can see that with growth of k, the gap between the fastest and the slowest receiver can become larger. The design of the in-network cache helps in absorbing the heterogeneity of the receivers.

Now we show that out-of-synch is difficult to avoid in a NDN router deployment.

Remark. *The problem of receivers going out-of-synch persists with receiver-driven feedback-based in-sequence congestion control protocols with heterogeneous receivers.*

The cache size at a router will inevitably be much smaller than the total amount of content available in the network. According to [9], NDN requires a 25TB cache for a 50% hit rate on Youtube data and 175TB cache for a 50% hit rate on BitTorrent data. However, [10] suggests that deployable NDN routers (with ∼$1,500 overall hardware cost) would likely have around 100Gb of cache at current costs. Thus, a router cache will be much smaller than the required cache size for the kind of content accessed in current day networks.

For tractability, we assume that concurrent flows in a router share the 100Gb cache size equally. The relationship between the bandwidth ratio (B_L/B_H), file size (F) and number of flows is shown in Fig. 4a. The intersection of the curve with XY plane is where C=F (# of flows=$100Gb/F$). The region behind the curve represents the region receivers are out-of-synch. Note that both the X- and Y-axis are log-scale, which means the area when the receivers are in-synch is just a very small portion of the overall region.

Although a core router might have relatively larger cache, the number of concurrent flows on that router is also correspondingly large. The available proportion of cache for each flow is therefore still quite small. If a set of receivers request 20M bytes of data through a core router with 100k concurrent flows, the ratio of rates should satisfy the following:

$$B_L/B_H \geq 1 - (100Gb/100k)/160Mb = 0.99375$$

If B_H can reach 100Mbps, B_L should be ≥99.3Mbps. When we apply the hierarchical tree model in Theorem 2.1, the minimum download rate *vs.* level is plotted in Fig. 4b. Even with 50 levels in such a hierarchy, the minimum required download rate is still >73Mbps. This is difficult to achieve due to the number of flows multiplexed on a given link.

In this model, we assume that all the consumers start to request a same piece of content at the same time. We argue that this assumption is reasonable, especially considering the set of popular content items. With a reasonably large

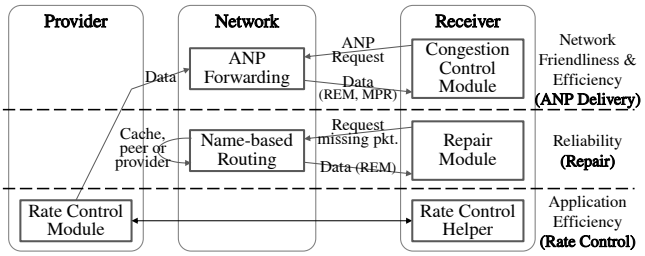


Fig. 5: SAID overview.

library of content accessed by consumers, the working set of popular content across consumers can potentially exceed the cache size at routers. It has been observed in [11] that there is a 'fat middle' with a large number of content items having multiple users accessing content at *around the same time*, even when unicast is used for VoD delivery (as is currently the case in the Internet with delivery of video over HTTP, similar to the request/response paradigm of several ICN protocols). Therefore, the best case scenario for normal query/response (sequence-specific requests) to avoid out-of-synch is when all the consumers request for the content at the same time (our assumption) and it would only worsen with the arrival of late-comers. Even with popular content, as larger numbers of heterogeneous users access the content, it will still result in cycling the cache (the symptom of 'out-of-synch') since the cache size at a router is likely to be much smaller compared to the working set size.

3. SAID FRAMEWORK

To address the out-of-synch issue at its root cause, SAID slightly modifies the request paradigm – from request for a specific-sequence to a request for *any-next packet* (ANP). The consumer-driven, network-assisted ANP delivery is the core of the SAID architecture (see Fig. 5). It intentionally decouples reliability from congestion control because *1*) the need for reliability and in-order delivery varies across applications, and *2*) the solutions that enforce in-sequence reliable delivery result in overheads (*e.g.*, receivers being out-of-synch, with NDN/ICP) or inefficiency (sending at the slowest receiver's rate, with pgmcc). The new paradigm seeks to retain consumers in-synch so as to maximize the utility of the packets sent at the first attempt. ANP delivery also ensures fairness on each branch of the dissemination tree, independent of the content provider's sending rate.

To satisfy applications that have different requirements on reliability and efficiency, we add application-specific modules to make SAID a general framework for multiparty information delivery in the ICN/NDN context. The repair mechanism (middle row in Fig. 5) fully utilizes name-based routing to help recover missing packets from the original provider, an in-network cache or neighbors, without sacrificing privacy or trust. For applications that have elastic bandwidth requirements (*e.g.*, file delivery), the rate adaptation mechanism (bottom row in Fig. 5) allows applications to achieve a tradeoff between network load and session completion time.

3.1 ANP Delivery Model

To overcome the case of heterogeneous receivers getting out-of-synch, and maximize the likelihood of delivery of a packet in the first transmission attempt, we propose a slight modification to the existing communication model of receivers asking for a sequence-specific packet (as is currently done by NDN and even TCP, *etc.*). Here, receivers ask for *any* subsequent incoming packet. Therefore, while con-

sumers with higher available bandwidth (referred to as faster consumers) can receive sequences 0, 1, 2, . . . , 100, consumers with lower available bandwidth (slower consumers) might get 0, 3, 6, . . . , 100. Further, consumers that join the group later than the others (we refer to them as late-comers) might get 66, 67, . . . , 100. We call this kind of delivery as in-synch delivery since all the consumers reach packet 100 around the same time. While it is true that the slower consumers still have to get packets 1, 2, 4, 5, . . . , and the late-comers have to get packets 0, 1, . . . , 65 (through repair, see §3.2), SAID ensures that in these cases, 1/3 of the packets are delivered in the first attempt, unlike sequence-specific requests that require retransmission of many more packets when receivers are out-of-synch. Thus we avoid cycling the cache.

An ANP request packet looks much like a normal Interest packet, except that the Name in the packet is the prefix of the file/flow instead of a sequence-specific content (*i.e.*, such as a normal Interest which carries both prefix and segment ID). To retain the current framework of NDN as much as possible, we reuse the existing Interest packet, since the name in the packet can be used for both sequence-specific and ANP requests. To distinguish between the two requests, we place an extra field (flag) into the packet header.

A pure network-based congestion control solution that divides the outgoing link bandwidth fairly for each flow going to the next hop is unable to accommodate indications of reduced demand from flows that have a limit downstream, especially with heterogeneous receiver bandwidths. This results in inefficiency, as was suggested by [12], and we show it in §5.1. Therefore, SAID chooses to use an end-system assisted (receiver-driven) solution to avoid such inefficiency. First, SAID retains the 'flow balance' property suggested in NDN, in which every ANP request packet will have at most one Data packet returned. Receivers then maintain a window of unsatisfied ANP requests, reflecting the maximum number of packets that can be in flight towards this receiver. We adapt this window using an AIMD scheme, just like TCP, thus finding the capacity of the bottleneck in the sender-receiver path.

The forwarding rules for ANP requests also need to be modified. One critical difference is that the ANP requests for a given flow would carry the same flow name (prefix). Therefore, instead of creating an entry in the PIT for each sequence-specific request, we add a counter in the PIT entries (called the Pending Request counter, PR) to keep track of the (maximum) window size of each downstream node. On receiving an ANP request, the router would: *1*) Skip the usual check of the Content Store to avoid transmitting duplicate packets. *2*) Check in the PIT if there is already an entry with the same name (prefix) and incoming face. If the entry does not exist in the PIT, add one entry with PR=1, otherwise, increment the PR by 1. *3*) If the PR value in the last step (before increment) was already the maximum PR for that prefix among all the incoming faces, an ANP request will be sent upstream based on a FIB lookup. With this mechanism, each consumer can propagate its window size up until its branching point[4], and the maximum window size is accumulated at the provider. However, the provider does not have to send at the rate of the maximum window (equivalent to aligning the sending rate at the fastest consumer), since it might result in more traffic generated because of repair. We describe how the provider should adapt its sending rate in §3.3. This modification to the forwarding rule for ANP requests is more space-efficient than the sequence-specific requests. For n requests of the same flow, SAID only needs 1 PIT entry and 1 counter compared to n separate PIT entries with the existing NDN approach.

On receiving a Data packet of the flow (which satisfies the ANP request), the router would: *1*) Save the packet in the Content Store for further (sequence-specific requests). *2*) Find all the PIT entries the Data can satisfy and decide the outgoing faces. For ANP request entries, the router would decrease the PR of each outgoing face by 1, and discard the PIT entry when PR becomes 0. *3*) Replicate and forward the Data to all the outgoing faces determined in the previous step. With this subtle change, receivers in SAID can receive data without getting out-of-synch, thus enhancing network efficiency. Since the window size of each consumer is propagated up to its branching point, we can ensure that the consumers would *1*) receive the flow at the rate indicated by the window size if that is smaller than the sending rate of the provider, or else *2*) receive at the sending rate of the provider. In §4, we describe how we control the window size on the consumer to reflect the fair share on the bottleneck between each consumer and its branching point, and thus ensure the max-min fairness in the network.

3.2 Efficient Repair

For applications that need different degrees of reliability, we depend on the application- (or transport-) layer interface to retrieve missing packets. SAID allows the application to decide *which packets* to repair and *when* the repair packets are delivered, while the network helps to find *where* copies of the needed packets. For applications like live-streaming, they can start the repair during the time period when the flow's new packets are being sent for the first time, although that might aggravate congestion at the bottleneck. The benefit is that the user experience (in terms of stall time) will be improved. This is a tradeoff that the application has to decide, based on policy and/or periodically based on perceived QoE. QoE is also influenced by repair being made through other paths. For applications like file download, repair can start after an ANP round to minimize the network congestion while new packets are flowing. Applications can also decide which packets need to be repaired, *i.e.* fetched. *E.g.*, applications using Raptor Codes [13] can reduce the amount of repair since the content of some missing packets might be recovered by ANP delivery of future packets.

Once an application decides to repair a packet, SAID uses NDN's sequence-specific (normal) data retrieval capability to retrieve the missing packets from the network. Since in-network caches may have stored packets as they are forwarded, some of the repair requests can be satisfied by the caches. However, if the required packets are no longer in the cache (have been replaced), consumers can still get the packets from the original provider. However, this repair has the potential to aggravate the congestion at a bottleneck link in the path from the provider to the receiver. Therefore, we propose a peer repair mechanism (described below) that can potentially mitigate the impact of repair on an already congested link by making use of other paths.

With SAID, receivers who have received (some or all of the packets) the flow can propagate FIB entries for the pre-

[4]We define the branching point of consumer C (BR_C) as the router closest to C that has another consumer that has bandwidth higher than C. The branching point for the fastest consumer is up at the first hop router of the provider.

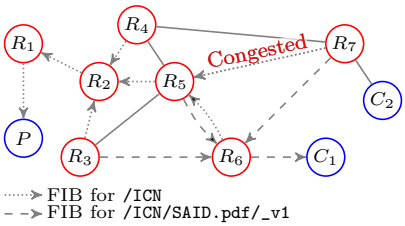


Fig. 6: Example of different repairs.

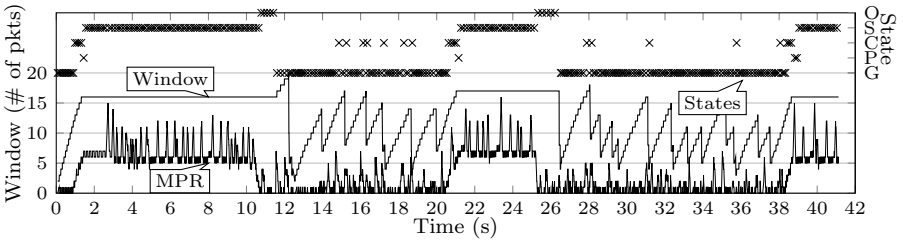


Fig. 7: Result of window control at the receiver using state machine.

fix of the flow over a limited number of hops. *E.g.*, in Fig. 6, after receiving the packets, C_1 propagates prefix `/ICN/SAID.pdf/_v1` over a 2 hop range. When C_2 requests for `/ICN/SAID.pdf/_v1/_s20`, R_7 will forward the request to R_6 and will eventually get a response from C_1. The repair can then bypass the congested link R_5–R_7 and still increase the overall useful throughput to C_2. Note that propagating FIB and the repair requests do not carry the identity of the consumers. Thus, the solution can fully utilize the benefit of NDN without sacrificing privacy and trust (the receivers and senders do not have to reveal their identity to one another). We believe multiple paths will continue to exist, just as we see in current networks [14]. In such a case, repair requests may be forwarded along the path that data was received successfully before, so as to increase the probability of receiving repair data (as proposed in [15]).

The simple mechanism we describe above might still be inefficient. We know that whenever C_1 propagates the prefix, routers will forward repair requests from nearby nodes to C_1 (instead of going upstream). Therefore, for efficiency, C_1 can only propagate the prefix when he receives *all* the packets in the flow, so as to be able to respond to the repair requests. Such a propagation is however useless for applications like VoD that needs repair while packets of the flow are still being delivered. An alternative solution for FIB propagation is for C_1 to propagate the exact ContentName of each packet he receives. Requests from slower receivers can then be redirected to C_1 immediately. This solution benefits applications that need 'in-flow' repair, but it places a large overhead on the FIB since every packet will result in a FIB entry in the nearby routers.

Therefore, to achieve a balance between the goals of repair efficiency and reduced FIB size, we suggest that the data provider should group n packets into a 'chunk' and replace the `segmentID` in the ContentName with `chunkID/segmentID`. *E.g.*, if n=100, the name of the packet with segmentID=205 should be `/ICN/SAID.pdf/_v1/_c2/_s205`. This will not affect the basic solution since every packet still has a globally unique ContentName. Nonetheless, C_1 can propagate `/ICN/SAID.pdf/_v1/_c2` after receiving packets 200-299 (*i.e.*, FIB entry for the chunk) either via the initial transmission or via repair. FIB entries will be smaller and at the same time will enable other receivers to get repairs from peers much earlier. The chunk size can be specified by the data provider based on the timeliness requirement for the flow.

3.3 Provider Rate Adaptation

ANP delivery ensures the fairness of a flow independent of the transmit rate of the sender. However, different applications have different transmission rate requirements, and the sender needs to adjust the sending rate accordingly. Live streaming and conferencing applications need a certain (minimum) sending rate based on desired video/audio quality. The sender can simply send data at that required rate.

Other applications may have more flexibility, but might have a preference to either satisfy most of the receivers in the first round of delivery or seek to satisfy the fastest receivers (*e.g.*, BitTorrent). For such applications, the transmission rate of the provider is a tradeoff between the (network, provider) load and the content delivery completion time. A higher sending rate could shorten the receiving interval on the faster receivers, but also result in higher retransmission rate in the network to reliably deliver to the other receivers. A lower sending rate may attempt to operate at the opposite end of the spectrum – sacrificing completion time for lower network load. As a selective component at the application layer, SAID also accommodates a mechanism to enable data providers to determine this balance. Instead of specifying a constant sending rate, we allow the provider to pick an ACKer from among the receivers to pace the flow similar to pgmcc [1]. The ACKer would send an ACK (*e.g.*, in the form of an Interest) back to the sender for each Data it receives. A feedback loop would be established between the ACKer and the sender like TCP and therefore the rate of the sender is aligned to the available bandwidth of the ACKer. However, unlike pgmcc, the ACKer does not have to be the slowest receiver. Please refer to our technical report [16] for details on ACKer selection.

4. CONGESTION CONTROL FOR ANP

With SAID, since the sending rate is not controlled by the window of each receiver, the difference between the sending and receiving rate can result in the receiver missing some of the packets in the sequence (we use the term 'holes in the sequence'). With SAID, these holes should not be treated as an indication of congestion (as conventional TCP does with loss-based congestion indication), triggering a response from the receivers. Instead, SAID uses REM [17] in a modified manner to indicate congestion appropriately. We also make the important observation that because the sending rate is not controlled by the receivers' congestion windows, receivers with an available bandwidth higher than the transmission rate will not see congestion (or packet loss). Their window will continue to grow (not unlike an uncongested TCP flow's window). This growth in the window essentially causes a large pending request (PR) value at upstream routers. A large PR value will allow a large number of packets to be delivered in a burst towards the receiver (*e.g.*, when the provider decides to increase the transmit rate or when upstream contention on the path disappears). This large burst of Data that will be queued at the bottleneck router (with consequent increase in feedback delay) causes that router to indicate congestion (*e.g.*, REM) on a large burst of packets. This can cause the receiver to over-react and reduce the window too much, thereby ceding more than its fair share to a competing flow. Subsequently, the competing flow would have a large PR, and now in turn experience a similar burst (a 'bang-bang' effect). Although

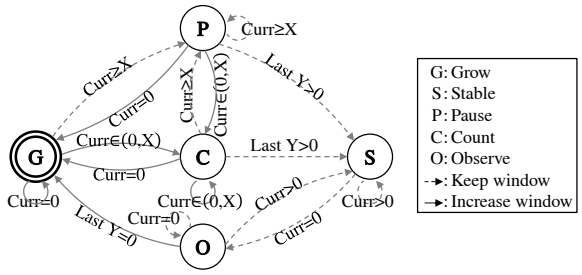


Fig. 8: State machine for receiver window increase.

the AIMD mechanism will eventually converge, the time it takes to converge will depend on the size of burst and feedback-delay. This effect therefore has the potential to under-utilize the network and/or cause unfairness between flows for a subsequent interval. While an excessively large window might cause inefficiency and unfairness, a very small window might also lead to inefficiency because a short burst caused by transient congestion might consume all the pending requests and result in unnecessary packet loss. Because of these considerations, we develop additional mechanisms to achieve congestion control and fairness.

4.1 Identifying Congestion

Since holes in the sequence are no longer an indication of congestion, we depend on help from the network to indicate congestion through Active Queue Management (AQM) mechanisms. In particular, we use Random Early Detection (REM [17]) in our current implementation. The solution can also take advantage of other AQM mechanisms like CoDel [18] that marks the packets on seeing congestion.

With REM, a simple AIMD mechanism on the receiver side (additive increase on receiving data and multiplicative decrease on seeing marks) works acceptably, especially on receivers that have available bandwidth below the sending rate of the provider. The introduction of AQM into NDN can also simplify existing congestion control solutions and help avoid any unnecessary bufferbloat [19].

4.2 Handling Additive Increase

In order to avoid the 'bang-bang' effect and optimize network utilization, the optimal window size should be the bandwidth-delay product of the path between the receiver and its branching point. It can vary depending on the network status, and can vary over time. Therefore, we propose a dynamic mechanism that observes and reacts to the minimum pending request count (MPR) on the path, with a goal of keeping the window size close to the optimal value.

The network piggybacks the instantaneous MPR value with data packets forwarded to receivers. Note that different receivers can see different MPR values even when they receive the same data packet since MPR is set at the branching point. A receiver observes the MPRs over a window so as to: *1*) get the minimum MPR value in that window (MMPR), thus avoiding excessive pending requests in the path, *2*) ensure that MMPR is within a small value but larger than 0 to avoid all pending requests being consumed by a future short burst, and *3*) smooth out variations. To simplify the calculation, our approach decides whether the window should increase based only on MMPR.

We use a state machine (depicted in Fig. 8) to decide the growth of the window. When MMPR==0 (state G), it implies that all the requests are consumed by the Data (incoming Data rate is less than the bandwidth-delay product of the downstream path) and therefore the receiver should increase the window size. With the growth of the window size, the MMPR would grow above 0 (state C in figure). The receiver should keep increasing the window size until he sees MMPR>0 for Y times (state S), which means the window size is big enough and stable. During the increase phase, if the MMPR grows above X (state P), it implies that there is a sudden change in the network and that the window size might be big enough already. The receiver would then stop increasing the window size immediately and wait for the status to either go back to G, C (increase again) or reach S (stop increasing). MMPR might get back to 0 when the receiver is in state S due to the change in the network status, and the receiver should increase the window again. However, to prevent instability in the network, we decide to increase the window only after seeing MMPR=0 for Y times (state O). Note that no matter what state the consumer is in, once it receives a marked packet (indicating a congestion), it will reduce the window size by half, go back to state S and start increasing the window. According to our experiments (in both emulations and simulations), we find it reasonable to have X=Y=5. We see that with these values for X and Y, the mechanism limits the window size properly and avoids bursts when competitors arrive (the bang-bang effect).

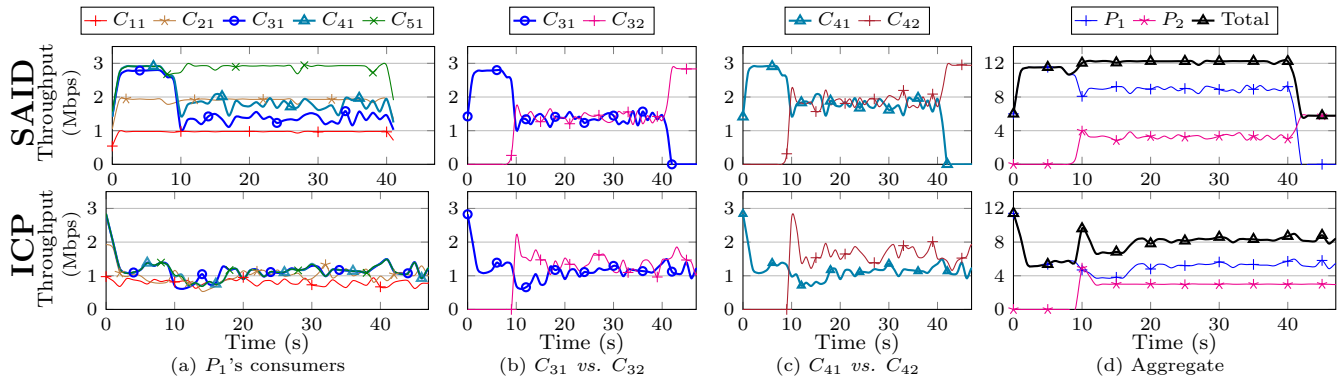
We use an example to trace the MMPR and the window increase decisions to aid in understanding how the scheme works with the values for X and Y mentioned above. The MPR, window size and the state transitions in Fig. 7 are shown for a receiver with $2Mbps$ bandwidth. Every mark represents the state of the state machine (corresponds to Fig. 8) at the end of each window. The MMPR grows in accordance with the window size after $1sec$. After 4 windows (decisions), the MMPR reaches 5 and the receiver stops increasing the window and MMPR stays around 5. Although the MMPR reduces below 5 for several windows at around $4sec$ (due to transient changes) the window size is not increased. When there is new contention at the bottleneck link (an additional competing flow) occurring between $10sec$ and $25sec$, MMPR drops to 0 and the receiver then begins to increase the window. When the receiver receives marked packets (REM) at $12sec$, it uses AIMD to respond to the congestion introduced by the competing flow. We show how this mechanism aids in achieving fairness in §5.1.

5. EVALUATION

We first present evaluations based on our prototype implementation. We then present simulation results for two applications (video streaming and large data file delivery) on a small custom topology (shown in Fig. 10) as well as a large RocketFuel Topology [20] with heterogeneous receiver bandwidths in our custom simulation environment (used in our previous work [21]). We compare SAID with *1*) An ICP [5] solution enhanced to use REM (so as to provide a reasonable comparison with SAID); and *2*) pgmcc [1], a window-based multicast congestion control protocol which aligns the data sender's transmission rate to the slowest receiver.

5.1 Emulation of ANP Delivery

We implemented the core components: *1*) ANP delivery (described in §3.1), and *2*) congestion control (described in §4) in Linux. The modified forwarding engine is implemented in user space and packets are encapsulated in UDP, as with CCNx. Our testbed consists of SAID enabled routers, each running on a single machine (CCNx 0.4). All the consumers and providers are run on a single machine so



(a) P_1's consumers (b) C_{31} *vs.* C_{32} (c) C_{41} *vs.* C_{42} (d) Aggregate

Fig. 9: Emulation result in competition scenario (using topology in Fig. 10, X=10$Mbps$).

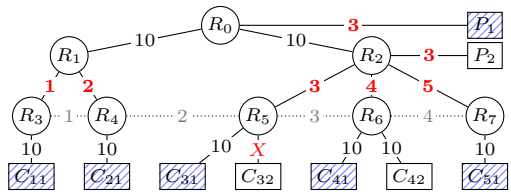


Fig. 10: Dissemination tree topology (bandwidth in $Mbps$).

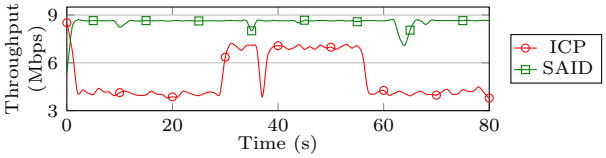


Fig. 11: Aggregate throughput in 4-consumer scenario.

as to have a synchronized clock for accurate result collection. The baseline topology consists of 8 routers (R_0→R_7), 7 consumers (C_{ij}) and 2 providers (P_1, P_2) as illustrated in Fig. 10. Per link latency is 2ms and the numbers on the links represent link bandwidth in Mbps. The bottleneck bandwidth for the consumers are marked in bold red.

• Efficiency of ANP Delivery Model

Fig. 11 illustrates the aggregate throughput achieved by SAID and ICP in the presence of 4 consumers. pgmcc aligns to the slowest subscriber, and its results are easily obtained without needing simulations[5]. The experimental setup was used in §2.2 with consumers C_{11}, C_{21}, C_{31} and C_{41}, having bottleneck bandwidths of 1, 2, 3 and 4 Mbps respectively and the content provider sending at 3Mbps. SAID is able to achieve an aggregate rate close to the maximum achievable throughput (1Mbps (C_{11}) + 2Mbps (C_{21}) + 3Mbps * 2 (C_{31}, C_{41})) of 9Mbps. ICP's maximum throughput is about 4Mbps. With ICP, receivers go out-of-synch and compete with one another on the link between P_1 and R_0. pgmcc also achieves around 4Mbps (1Mbps * 4) throughput.

• Fairness in the Presence of Competition

The experimental setup is P_1 with 5 consumers (named C_{i1} in Fig. 10) and P_2 has 2 consumers (C_{32}, C_{42}). To evaluate SAID's fairness when there is a competing flow from another provider (P_2), we configure P_2 to start sending packets after approximately 10s, with a sending rate of 3Mbps (*i.e.*, as fast as it can) and study its influence in the network under an extreme case. C_{31} and C_{32} are competing for the bottleneck bandwidth between R_2 and R_5 (3Mbps) and C_{41} and C_{42} are competing for the bottleneck bandwidth between R_2 and R_6 (4Mbps). The bandwidth between R_5 and C_{32}, *i.e.* X, is set to 10Mbps so that the link between R_2 and R_5 is the bottleneck link.

Fig. 9 illustrates the throughput achieved at each receiver. As in Fig. 11, we observe in Fig. 9a that in SAID, each of P_1's receiver is able to receive at the bottleneck capacity until 10s. Once the competing flow arrives, the receivers in SAID are able to receive at their *(statistical) fair share* of the bottleneck bandwidth. In ICP, upon the arrival of a competing flow, the throughput achieved by each receiver drops significantly, from an average of 4Mbps to just 1Mbps.

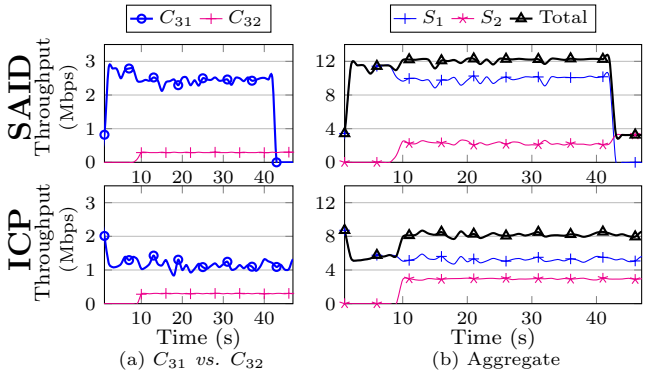
Fig. 9b–9c show that in SAID, both C_{31} and C_{41} are able to receive at their individual fair share (1.5Mbps and 2Mbps) in the presence of competition from P_2's flow. On the other hand, with ICP, the consumer's of both P_1 and P_2 are not able to fully utilize their fair share of the bottleneck bandwidth, resulting in an under-utilization of the bottleneck bandwidths. This is due to the fact that both C_{31} and C_{41} are in fact competing with other consumer's from P_1 in the P_1 to R_0 link and C_{32} and C_{42} are competing with each other in the P_2 to R_0 link, due to the out-of-synch phenomena. The peak at the start of the P_2 flows in ICP, illustrate that for a short period, flows to C_{32} and C_{42} are in-synch.

In Fig. 9d, the aggregate throughput achieved by P_1's and P_2's consumers is close to the maximum achievable throughput with SAID. Till the first 10s, when only P_1 is active and the total achievable throughput of the network is 12Mbps, we observe that SAID is able to get closer to this rate. When P_2 starts at 10s, the ideal achievable aggregate throughput increases to 13Mbps. SAID is able to get closer to this ideal rate. With ICP however, it is approximately 5Mbps before 10s and 8Mbps afterwards. In pgmcc, in the absence of a competitor (*i.e.*, before 10s), P_1 aligns to C_{11} (1mbps) and therefore the total throughput is 5Mbps (all its 5 consumers receive at 1Mbps each). In the presence of a competitor P_2 (after 10s) who is sending at 2Mbps (aligned to C_{32}), the total throughput will increase to 9Mbps.

• Benefit of Receiver-Driven Mechanism

We now modify our experimental setup to a simple dumbbell topology where the link bandwidth between R_5 and C_{32}, *i.e.*, X is 0.3Mbps. Since C_{32} can receive only at 0.3Mbps from P_2, C_{31} is able to make use of the spare capacity of its bottleneck link between R_2 and R_5 and receive at approximately 2.7Mbps (see Fig. 12a), thereby making optimal use of that link (total usage is 2.7Mbps + 0.3Mbps). On the other hand, a flow based fair queuing approach would only achieve 1.8Mbps (1.5Mbps(C_{31})+ 0.3Mbps (C_{32})), since the link between R_2 and R_5 would be shared equally between flows from P_1 and P_2, only for P_2's packets to be dropped at R_5. This result highlights the benefit of using a receiver-driven approach for SAID. In the case of ICP, C_{31} is only

[5]We only implemented pgmcc in the simulation



(a) C_{31} *vs.* C_{32} (b) Aggregate

Fig. 12: Emulation result in competition scenario (using topology in Fig. 10, X=300$kbps$).

Table 1: Stall time (s) in streaming demo (video length=40s).

	C_{11}	C_{21}	C_{31}	C_{41}	Rep.
Baseline	83.384	22.507	2.461	0.886	–%
ICP	90.530	33.965	33.821	33.820	–%
pgmcc	84.804	84.770	84.768	84.767	0.00%
SAID-F	83.821	40.062	39.569	1.131	12.44%
SAID-S	83.541	22.754	4.010	1.123	21.91%
SAID	44.304	1.271	1.151	1.131	12.44%

able to receive close to an equal share of the R_2 and R_5 link bandwidth (<1.5Mbps) because the receivers are out-of-synch. ICP gets approx 1.8Mbps aggregate throughput on that link. The total throughout for SAID (12 Mbps) and ICP (8 Mbps) are shown in Fig. 12a. For pgmcc, in the presence of a competitor, the total achievable throughput is 5.6Mbps (1Mbps * 5(C_{11}-C_{51}) + 0.3Mbps * 2(C_{32}, C_{42})).

5.2 Simulation of different applications

• Video Streaming:

We consider a streaming video application with a play-out rate of 3$Mbps$. We evaluate the effectiveness of the approaches using the *Stall time*, which reflects the impact on user experience. No stalls occur when there are no holes in the sequence in the play out buffer for the next 1s of video playback. For the baseline, we run the simulation 4 times with a single receiver requesting the video. SAID-F is when repair is by the provider at the end of the flow (similar to 'download-and-play'). SAID-S is repair by the provider while streaming the video. SAID is the peer-assisted repair solution we propose here. These solutions are compared on the tree topology (Fig. 10, with only C_{11}-C_{41} activated) as well as ICP and pgmcc.

Table 1 shows the stall time for each consumer and the repair ratio (*i.e.*, the percentage of the total number of packets received via repair over the total number of packets received, shown as 'Rep.' in the Table) for a 40s video. In ICP, when all receivers request the video simultaneously, they go *out-of-synch* soon thereafter. The stall time becomes larger than the baseline especially for faster receivers. With pgmcc, since the provider has to align with the slowest receiver (C_1), the stall time for the rest of the receivers is up to 80s larger compared to the baseline. The user experience for the faster receivers therefore deteriorates considerably.

Although SAID-F achieves a relatively low repair rate, the stall time for C_1-C_3 remains high, because the repair is performed after the flow finishes. C_3 also experiences loss, even though its bandwidth is equal to the playout rate. This happens due to the variability in the network and the overhead of packet headers. The exact receive rate for the application can be a bit lower, despite C_3 having a nominal 3$Mbps$ available bandwidth. C_2 and C_3 benefit from the repair during streaming in SAID-S, but this benefit comes at the cost of higher network load. The repair ratio of SAID-S is higher than SAID-F (∼22% *vs.* ∼12.5%) since the retransmission has to go through the bottleneck link and affects the primary 'any packet' stream. Nonetheless, the retransmission rate from the provider of SAID-S is still lower than ICP (∼1.9 *vs.* ∼3.4, not shown) which means SAID-S consumes less network and provider resources. SAID (our proposal) however is superior as it is able to utilize the extra bandwidth between end-hosts. The repair does not affect the multicast session and the slower consumers can get twice the bandwidth compared to SAID-F and SAID-S. Despite the repair, the stall time for the slower receivers (C_1−C_3) is much smaller than with the other solutions, even though they are playing the video at the same rate of 3$Mbps$. We varied the video length and observed that the stall time grows proportionally with the video length but the pattern in Table 1 still holds.

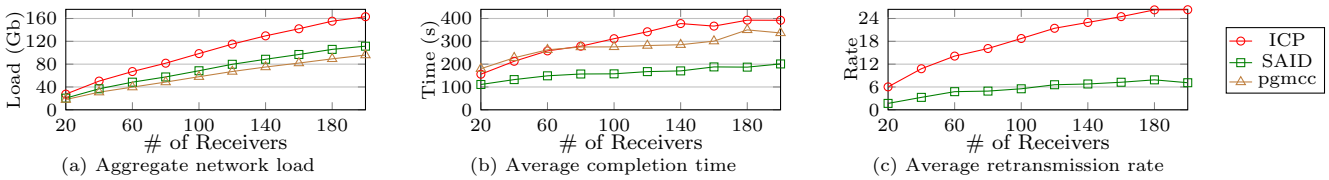
• File Content Delivery:

We evaluated the complete SAID solution with the RocketFuel topology (AS-3967). We randomly place 20-200 receivers on the 79 core routers. Since RocketFuel does not have bandwidth information, we assign available bandwidths in the range of 1-10$Mbps$ for each link. The result of a trace with 100 flows using ICP, SAID and pgmcc is in Fig. 13.

By decoupling reliability from congestion control, SAID has lower network load compared to ICP (Fig. 13a), especially as the number of receivers increases, by up to 46% with 200 receivers. Since ∼60% of the data is delivered at the first attempt, SAID has a much lower average # of transmissions of each packet from the provider and average flow completion time compared to ICP (Fig. 13c). However, SAID consumes more network bandwidth (∼10%) compared to pgmcc since it aligns to a faster receiver and uses repairs subsequently. For the 10% additional network load, SAID achieves lower average completion time (by ∼65%, Fig. 13b).

6. RELATED WORK

Many reliable multicast protocols have been proposed to enable large scale reliable data dissemination. To maximize the utility of multicast, cyclic- and scheduled-multicasts [22, 23] have been proposed to benefit the consumers that are not starting at the same time. These solutions can be broadly classified into two categories based on the chosen repair mechanism: provider repair or peer-assisted repair. The clients in provider-repair mechanisms like [24,25] send NAKs to the provider (or the whole group to suppress duplicate NAKs) and the provider retransmits the missing packets specified in the NAKs. The data provider in such solutions has to align the sending rate to the slowest receiver eventually. Peer-assisted approaches like [26,27] group receivers in a hierarchical structure and the receivers ACK to the upper level in the tree so that the ACKs can be aggregated. By introducing such a relationship among receivers, these solutions allow receivers to perform local repair and therefore, the provider can align the sending rate to the majority of (or the fastest) receivers based on the ACK strategy. Unfortunately, in these proposals, the subscribers have to exchange information in a peer-to-peer manner in the IP network to perform repair as well as send ACKs. According to [28], in these solutions the receivers have to reveal their identities (IP addresses) to the peers, and thereby trust them as there is no guarantee of data integrity during peer-repair.



(a) Aggregate network load (b) Average completion time (c) Average retransmission rate

Fig. 13: Simulation result for file content delivery application.

Layered multicasts are also proposed to deal with heterogeneous consumers. In [29], the provider creates different multicast groups that transmit different resolutions of the data. The receivers can select appropriate groups according to their link capacity. These solutions are applicable to select applications. Nonetheless, having a reliable multicast capability for a single rate stream is still fundamental for broad-based use across all kinds of applications. Therefore, the layered multicast solutions can be considered as orthogonal to the single-rate reliable multicast.

Pub/sub based large scale data dissemination solutions such as [21, 30] lack an efficient mechanism to ensure reliability and avoid congestion collapse in the network.

7. CONCLUSION

Designing a congestion control mechanism for efficient information delivery to large numbers of receivers is difficult, particularly with heterogeneous receiver bandwidths. Through emulation and an analytical model, we showed that heterogeneous receivers will get out-of-synch with existing receiver-driven, in-sequence request-response approaches. To overcome several of these limitations, we proposed SAID, a control protocol that allows receivers to request 'any-next' packet instead of the 'next in-sequence'. SAID retains network fairness and delivers more packets on the first attempt. To be a general-purpose control protocol that satisfies different application requirements, SAID also provides application-specific modules. The first is a repair module that exploits content at peers and in-network caches to provide varying degrees of reliability to match application need. The second is a provider rate adaptation module that strikes a balance between session completion time and network traffic.

Our evaluations show that SAID achieves efficiency and fairness on each path between the provider and receivers. From a large scale simulation we show SAID reduces the aggregate network load (by ∼46%) and transmission completion times (by more than 50%) compared to ICP. SAID also reduces average completion time by ∼40% while only increasing network load by ∼10% (for repair) compared to pgmcc, which aligns the sending rate to the slowest receiver. Further, based on measurements on a prototype, SAID outperforms ICP, getting 50% higher aggregate throughput and almost twice the throughput of pgmcc. With the efficient repair of SAID, streaming applications have a much smaller stall time compared to other mechanisms.

8. ACKNOWLEDGEMENTS

This work is supported by the EU-Japan ICN2020 Project (EU HORIZON 2020 Grant Agreement No. 723014 and NICT Contract No. 184); and US NSF under Grant No. CNS-1455815. We thank our shepherd, Lixia Zhang, for her insightful feedback, and Xu Chen for his help on modelling.

9. REFERENCES

[1] L. Rizzo, "pgmcc: a tcp-friendly single-rate multicast congestion control scheme," in *SIGCOMM*, 2000.

[2] M. Castro *et al.*, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *JSAC*, pp. 1489–1499, 2002.

[3] V. Jacobson *et al.*, "Networking Named Content," in *CoNEXT*, 2009.

[4] L. Zhang *et al.*, "Named Data Networking (NDN) Project," PARC, Tech. Report NDN-0001, 2010.

[5] G. Carofiglio *et al.*, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *ICN*, 2012.

[6] S. Salsano *et al.*, "Transport-layer issues in information centric networks," in *ICN*, 2012.

[7] L. Saino *et al.*, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *ICC*, 2013.

[8] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM*, 1988.

[9] S. Oueslati *et al.*, "Flow-aware traffic control for a content-centric network," in *INFOCOM*, 2012.

[10] S. Arianfar *et al.*, "On content-centric router design and implications," in *ReArch*, 2010.

[11] V. Gopalakrishnan *et al.*, "Understanding Couch Potatoes: Measurement and Modeling of Interactive Usage of IPTV at Large Scale," in *IMC*, 2011.

[12] S. Floyd *et al.*, "Promoting the use of end-to-end congestion control in the internet," *TON*, pp. 458–472, 1999.

[13] M. Luby *et al.*, "Raptor Forward Error Correction Scheme for Object Delivery," RFC 5053, IETF, 2007.

[14] J. He *et al.*, "Toward Internet-wide Multipath Routing," *Network*, pp. 16–21, 2008.

[15] V. Sourlas *et al.*, "Information resilience through user-assisted caching in disruptive content-centric networks," in *IFIP Networking*, 2015.

[16] J. Chen *et al.*, "SAID: A Scalable and Adaptive Information Dissemination Protocol in ICN," Tech. Rep., 2014. [Online]. Available: https://www.net.informatik.uni-goettingen.de/publications/1912/SAID.pdf

[17] S. Floyd *et al.*, "Random early detection gateways for congestion avoidance," *TON*, pp. 397–413, 1993.

[18] K. Nichols *et al.*, "Controlling Queue Delay," *ACM Communications*, pp. 42–50, 2012.

[19] J. Gettys *et al.*, "Bufferbloat: dark buffers in the internet," *Communications of the ACM*, pp. 57–65, 2012.

[20] R. Mahajan *et al.*, "Inferring Link Weights using End-to-End Measurements," in *IMW*, 2002.

[21] J. Chen *et al.*, "COPSS: An Efficient Content Oriented Pub/Sub System," in *ANCS*, 2011.

[22] K. C. Almeroth *et al.*, "Scalable delivery of web pages using cyclic best-effort multicast," in *INFOCOM*, 1998.

[23] V. Aggarwal *et al.*, "The effectiveness of intelligent scheduling for multicast video-on-demand," in *Multimedia*, 2009.

[24] D. Towsley, "An analysis of a point-to-multipoint channel using a go-back-n error control protocol," *Communications, IEEE Transactions on*, pp. 282–285, 1985.

[25] K. Sabnani *et al.*, "Multidestination protocols for satellite broadcast channels," *Communications, IEEE Transactions on*, pp. 232–240, 1985.

[26] S. Floyd *et al.*, "A reliable multicast framework for light-weight sessions and application level framing," *TON*, pp. 784–803, 1997.

[27] S. Ratnasamy *et al.*, "Revisiting ip multicast," in *SIGCOMM CCR*, vol. 36, no. 4, 2006, pp. 15–26.

[28] N. Duffield *et al.*, "Distrust and privacy: Axioms for multicast congestion control," in *NOSSDAV*, 1999.

[29] S. McCanne *et al.*, "Receiver-driven layered multicast," in *SIGCOMM*, 1996.

[30] N. Fotiou *et al.*, "Developing information networking further: From psirp to pursuit," in *BROADNETS*, 2012.