2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI)

Wireless Multicast Rate Control Adaptive to Application Goodput and Loss Requirements

Mohammed Elbadry
Electrical and Computer Engineering
Stony Brook University
United States
mo.elbadry@ieee.org

Fan Ye
Electrical and Computer Engineering
Stony Brook University
United States
fan.ye@stonybrook.edu

Peter Milder
Electrical and Computer Engineering
Stony Brook University
United States
peter.milder@stonybrook.edu

***Abstract*—Modern IoT/edge applications require one-to-many wireless communication (e.g., multi-drone coordination, data sharing among vehicles, synchronized IoT light shows). Due to the constantly varying wireless medium, thus reception quality, the sender must adjust its transmitting rate on a per-frame basis to meet the goodput and loss requirements of multiple receivers. Deciding an optimal rate within tens of milliseconds from nearly a hundred or more choices and continuing to chase that moving target is extremely challenging. Existing wireless technologies have little support for multicast rate control: most works are designed for unicast, where one receiver sends explicit per frame feedback, which is infeasible to scale to multiple receivers; a few works for multicast have rigid structures and high overhead unsuitable for IoT/edge; and most designs are based on a common implicit assumption: higher rates incur more losses. In this paper, we conduct systematic experiments and find that only a small fraction of data rates are practically useful, and higher rates can incur similar or even lower losses, thus cutting the data rate table size by 3.8X, making it manageable to select the optimal rate within a short duration. We further design an application-adaptive multicast rate control feedback protocol (r-DACK) with two policies enabling receivers to specify their desired loss rate, or loss rate and goodput requirements. r-DACK enables most receivers to meet their goodput/loss requirements while not being "bogged down" by some stragglers with bad reception quality. We build a prototype leveraging 802.11ac radio hardware and show that r-DACK can meet various goodput (15-50Mbps) and loss rate (10-50%) requirements successfully, both indoors and outdoors.**

***Index Terms*—Wireless, multicast, rate control, WiFi, Edge networks**

I. Introduction

Wireless multicast is an enabling technology for many IoT/edge applications, such as a group of collaborative drones sharing images to plan subsequent trajectories for surveying or smart infrastructure (vehicles, road-side boxes, and pedestrians) sharing LiDAR/images for real-time 3D maps for transportation safety. Another example of multicast is a control console that sends commands to lightboxes distributed across a field at an outdoor concert. We need a wireless multicast rate control algorithm that selects a suitable transmitting rate to ensure high reception quality, in particular goodput and loss for receivers. Selecting a data rate requires configuring up to five parameters that together determine the transmission rate. Due to the fast-varying wireless medium, the reception quality at a receiver can change from frame to frame. Thus the sender must adjust its transmission rate constantly on a per-frame basis to ensure "optimal" reception. In typical existing wireless technologies like 802.11, a receiver sends explicit feedback (e.g., CSI [1]) to the sender for rate adaptation.

However, with multiple receivers, where each has a different reception quality and may require a very different transmitting data rate, determining the "optimal" rate becomes a challenge. Meeting all receivers' needs might be infeasible: some "stragglers" with bad reception quality may need an extremely low rate, thus "bogging down" everyone else. The difficulty in the decision is compounded by multiple per-frame parameters that determine the transmitting rate. In recent wireless communication standards such as 802.11n/ac/ax, the number of parameters has grown from two (modulation and coding) to five (adding guard interval, bandwidth, and spatial stream), increasing the available data rates from 20 to over 70 and even over 200 in newer standards. An incorrect selection will result in either high loss or low goodput. The decision also must be made extremely quickly, within tens of milliseconds, and be made again and again to keep up with the varying wireless medium.

Previous studies [2] have shown that large duration windows of exploration (e.g., seconds) can be misleading because medium changes happen at a much shorter time scale (e.g., tens of milliseconds). Thus the algorithm must select from the hundreds of options within tens of milliseconds for the optimal data rate across all receivers, and keep chasing that moving target that constantly changes. An intelligent searching methodology is needed to minimize the search duration. [*]

There are only a few rate control algorithms designed for multicast [3], [4]. They tend to have a high setup overhead where rigid structures among nodes (e.g., leader, cluster-based) must be established and known prior. Such designs are for stationary scenarios (e.g., light boxes); however, they do not work for mobile scenarios where peers do not conform to such structures (e.g., drone communications or vehicular multicast) and where the environment is ever-changing with nodes joining/leaving at any time.

Further, most existing works (especially those for unicast) have an implicit assumption: a higher data rate leads to more loss, and vice versa. Thus they decrease the data rate upon

[*]This work is supported by NSF grants 1652276 and 1730291.

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00007

higher loss, and increase upon lower loss. We conduct systematic experiments and find that this assumption does not actually hold: higher data rates may incur similar or even lower losses than some lower data rates, making them practically useless. Thus only a small fraction of data rates are necessary in selection. This was observed anecdotally in some earlier works [5] but no systematic study was performed. Other non-multicast works have considered machine learning techniques such as reinforcement learning to deal with the large data rate table size [6]. However, our discovery that only a small fraction of data rates is useful means such complexity is unnecessary.

In this paper, we design a multicast rate control algorithm that allows the sender to quickly determine a suitable transmitting rate for multiple receivers so as to meet their performance requirements (i.e., loss rate and goodput). We start with a systematic investigation on the five parameters that determine the data rate, and find new insights that reduce the usable data rate set in our real system implementation from 76 to 20 candidate rates (74% reduction), leading to a much simplified yet more effective design. To the best of our knowledge, we are the first to make the following contributions:

- We thoroughly investigate how the transmission rate determined by the five parameters correlates to loss in 802.11ac, based on over 2M packet traces totaling over 24 hours with four receivers in 12 different environments. We found that many parameter combinations transmitting at similar rates incur different losses. Thus, only a small fraction with the lowest losses is necessary, cutting the rate table size by 3/4. This breaks an implicit assumption made in many rate control works that consider all rates and decrease the rate upon more losses (and vice versa).
- These insights lead to a radically simplified yet highly efficient and practical design. We design a two-stage coarse and fine-grained search process where the highest impact parameters are selected first, and then we fine-tune the less impactful ones. We also design a multicast feedback protocol (r-DACK) that allows receivers to specify their desired performance requirements (i.e., loss rate and goodput, or loss rate only). Such requirements enable the determination of which consumers are "salvageable" and thus allowed to send feedback asking for retransmission to make up for losses. This avoids an "unsalvageable" consumer continuing to ask for retransmissions and thus slowing down everyone.
- We develop Linux kernel modules to build a real prototype running on 802.11ac dongles. Our results show that it achieves 30Mbps goodput indoors with multiple receivers and under 20% loss rate, and 50Mbps goodput and under 30% loss rate, whereas 802.11ac broadcast suffers above 80% loss rate, at a mere 6Mbps. Our algorithm costs less than 5% overhead of goodput, which is very little to provide retransmission feedback and adjust the data rate accordingly. The limit in goodput was due to the limited data rate table access in the dongle. With new radio hardware supporting access to the full data rate table (up to 2.3Gbps), we expect that r-DACK will achieve 1Gbps goodput at less than 20% loss.

II. Background

In this section, we provide background on: *i) Address and Content-centric Filtering*, describing the two paradigms of communications and their implications on MAC; *ii) 802.11 Parameters*, denoting all parameters in 802.11 regarding rate control and how they impact the medium; *iii) a Multicast DACK* protocol designed for V-MAC [7], a data-centric MAC layer where the consumer with the highest loss rate provides feedback asking the producer for missing frames after each round of transmission; *iv) Related Work* that has been done by the research community in recent years regarding rate control.

A. Filtering Paradigms

Filtering paradigms impact the knowledge that nodes have prior to the communication begins, which rate control can use to make a decision. *Content-centric* filtering (i.e., filtering based on content) enables multicast by nature where data can be sent to multiple consumers concurrently, and they can accept the data by checking their table (Pending Interest Table - PIT) for the content of Interest. A consumer requests data by sending an Interest with such *dataname*. This requires pre-defined prior knowledge of the *dataname* of content and mutual agreement between the consumer and producer. The *Content-centric* paradigm relies on a *Pending Encoding Table (PET)* which routes frames based on encoding (mapped to *dataname* in network layer).

Most MAC designs use an *address-centric* paradigm (i.e., filter based on destination ID) that relies on destination addresses in frames to filter unwanted ones and retain those needed by upper-layer applications. *Address-centric* does not enable multicast by nature. It requires some setup overhead when a group is formed, and a multicast ID is created where all interested receivers forward frames to upper layers carrying such ID as their destination.

A *content-centric* MAC uses content identifiers (e.g., data names) to filter and retain desired frames. Thus, the sender publishes on a "topic" without the need to designate any destination MAC addresses. Multiple frames on the same "topic" can be sent back to back to form a stream, during which the rate control algorithm can search and identify the suitable parameter combination for transmission. V-MAC [7] is so far the only content-centric MAC design. It has a feedback protocol (DACK) where the worst performing consumer reports its missing frames soonest to solicit retransmissions, which also recovers for others with fewer losses. V-MAC works well with applications that only care about loss rate. However, it does not work well with ones that care about goodput and may require a certain speed; the worst consumer may request excessive retransmissions, dragging everyone down.

B. 802.11 Parameters

In the 802.11g standard [8], the per-frame data rate can be adapted by changing the following parameters: *i) modulation* and *ii) coding rate* with maximum 54Mbps. With the 802.11n standard [9], two more parameters were introduced: *i) Guard Interval (GI)* enabling using different spacing between the symbols (800 and 400ns); *ii) Spatial Stream (SS)* enabling transmitting the frame over multiple streams concurrently within the band; *iii) Bandwidth* enabling increasing bandwidth (40MHz), making the maximum nominal data rate possible up to 300Mbps. In 802.11ac, no new parameters were introduced; however, the ranges of parameters further increased significantly (160MHz bandwidth, three spatial streams), resulting in the ability of radio to go up to 2.3Gbps. Thus, the number of parameter combinations determining data rate increased from 32 in 802.11n to 232 in 802.11ac. Such a huge space brings a great challenge: the algorithm must decide which subset of combinations to identify the best one without excessive latency.

With high data rates, we run into the problem of preamble overhead. The preamble is a bit pattern sent before any data frame to synchronize the sender/receiver antenna. A short preamble is always deployed with newer standards (96μs). However, standard 802.11 data frames usually have 1.4KB of payload, which takes a minimal duration when transmitting at a high rate (e.g., 28μs at 400Mbps, 29% of the preamble duration). Thus, most air time is used on the preamble, not the payload. This leads to the introduction of A-MPDU and A-MSDU [10] (first introduced in 802.11n) to bundle much larger data payload after one preamble to improve the air time utilization(up to 4.6MB in 802.11ac [11]).

C. Related Work

We cover both unicast and multicast rate control algorithms. The difference between unicast and multicast is how feedback is obtained from multiple receivers. However, the goal and rate selection can be the same.

Unicast algorithms rely on a one-to-one communication where a frame is sent, and an ACK is received before moving on to the subsequent frame transmission. There is a retry chain where a frame can be retried if no ACK is received multiple times. Current WiFi adapters adopt many existing algorithms that leverage such mechanisms. One of the most common algorithms is minstrel [12], which relies on a *lookaround rate* and average rate. Usually, most of the frames (90%) are transmitted using the average rate and much less (10%) using the lookaround rate, which opportunistically tries a higher rate to see whether it can improve.

Yin et al. [13] perform a thorough survey on rate control algorithms. We share the most common features and what is necessary for our work below. Besides all existing rate control algorithms being designed for unicast, there are common features to be noted among all algorithms: *i)* rate control algorithms rely on the status of frame transmission or reception of the last frame or adaptation window; *ii)* different metrics are deployed to estimate error or correlate to loss of frames for better performance (e.g., Bit Error Rate (BER), Channel State Information (CSI), Signal to Noise Ratio (SNR), etc.); *iii)* most designs follow a "ladder" up or down to adapt rates, assuming a higher rate has a higher loss. Our studies disprove such an assumption where we find that higher data rates do not necessarily increase the loss rate (section IV-A), calling into question such designs.

Many mechanisms have been designed over the years for different standards. Throughput-based algorithms (e.g., MIRA [14], STRALE [15]) aim to maximize the throughput through analysis of current throughput and predict how other rates will perform in terms of throughput. Another approach is Consecutive Transmission Result (CTR) algorithms (e.g., CARA [16]), which require RTS/CTS and use a busy counter. However, these mechanisms look into the data rates, organize them based on throughput, and do a ladder (either one or multiple rates up/down). Some works have attempted using machine learning (reinforcement learning [17]), and this approach shows promise, justifying the need for ML due to the large set of options to choose from. These works are for unicast (including the ML one) and rely on one frame-ack feedback style.

Multicast Work. Most of the works rely on collecting aggregate feedback from receivers. Some works integrate Automatic Repeat Request (ARQ) mechanisms [18] and others utilize RA methods [19]. There are two essential feedback mechanisms:

i) Leader-Based, where a leader is selected to either acknowledge the frame [18] or negative acknowledgment [19] where the frames that are not received are reported; *ii) Cluster based* [20], where receivers are partitioned into clusters, and the receiver with the weakest link is selected to report. Both *(i)* and *(ii)* require having a designated receiver(s) that gets selected in some fashion, which we cannot afford given the loose formation of our application. Neither case guarantees reliable reception across all receivers nor allows the receiver to try to request missing data.

All multicast works we have surveyed require an overhead where either the transmitter, the receivers, or both must know all those involved before and during the transmission to designate roles; such approaches form rigidness that provides performance benefits but hinder sloose network operation for our application needs.

III. Design Overview

We divide this section into the following: *i) Goals and Assumptions* listing our design requirements that need to be met for a successful system; and *ii) Challenges* describing the three challenges, which we solve with three components to achieve multicast rate control.

A. Goals and Assumptions.

We summarize our goals as the following: *i)* support applications' flexible needs (e.g., file transfer and real-time video) on edge; *ii)* allow multiple consumers to participate in multicast rate control without prior coordination to support nodes joining and leaving dynamically; *iii)* converge to the optimal rate as fast as possible. The optimal rate is the maximum goodput across all consumers while meeting loss rate requirements. We also make the following assumptions:

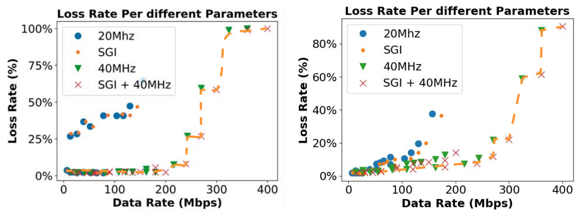
i) some consumers can hear each other and take actions (e.g., canceling a feedback transmission); *ii)* consumers and producer agree on the desired policy's performance thresholds before the beginning of communication. With these assumptions, we can design a system where consumers can hear each other's communication with the producer and make decisions (e.g., do not send feedback asking for a missing frame if another consumer already reported it), with consumers and producers agreeing on the exact requirements, a producer can make intelligent decisions during transmissions to provide best application performance (e.g., do not retry transmitting x lost frame(s) as they are not necessary due to application layer coding); *iii)* no setup overhead allowed nor prior knowledge between consumers and producers about each other is available (to allow for loose network communications where consumers can join and leave throughout transmissions).

B. Challenges

In 802.11ac, there are five parameters to configure, resulting in a data rate. The search space is large, and there is no known method to determine how to order them (if they can be). After understanding the parameter space and navigating within, we need to find how to select the rates within the space (which parameter to change, when, and why?). Lastly, given the nature of multicast rate control, we need a loosely coordinated method where consumers can provide feedback on their loss and different rates of performance where feedback frames do not overwhelm the producer with information and saturate the medium. The three main problems are addressed individually in Section IV-A *Parameter Space Exploration,* Section IV-B *Rate Selection*, and Section IV-C *r-DACK*.

Through long-term diverse data collection, we observe that higher data rates do not necessarily correspond to higher loss rates, and that only a small fraction of rates (and their parameter combinations) are needed. Through such findings, we reduced the search space by finding that some data rates throughout all environments and duration of experiments incurred higher loss rates than others with higher goodput. Besides eliminating data rates and reducing the search space, we comprehensively analyzed all rate control parameters (i.e., Modulation, Coding Rate, Bandwidth, Guard Interval, and Spatial Stream (SS)). We find that modulation and spatial stream must be correct first for proper performance, followed by coding rate and guard interval. We dive into our analysis in Section IV-A.

We must select the correct modulation and spatial streams followed by coding rate and guard interval based on our parameter exploration findings. We design a coarse-then-fine-grained search strategy that can quickly identify the optimal rate without any degradation in wireless performance. We designed a performance policy structure that allows applications to provide input on rate selection. This structure enables applications to determine their desired multicast loss rate and goodput thresholds, which are then used by the MAC to determine which receivers can be "salvaged," i.e., retransmissions can meet its goodput and loss thresholds. A receiver can be salvaged when it is possible to reduce its loss rate below the policy's requirement and fulfill the goodput requirement. Besides defining which receivers are worth saving, the performance policy affects which data rates to select. Our design supports two policy types: a joint loss rate and goodput policy, and loss rate-only policy. Section IV-B details all the structures designed for multicast rate control and how rate selection is done.



(a) Corporate (b) Home

Fig. 1: The loss rate of all data rates in two different environments. Data rates that connect with the dashed line are Pareto optimal.

We have also developed a rate control DACK (r-DACK) feedback algorithm that takes into account performance thresholds. This algorithm selects the receiver that requires the most retransmissions while ensuring that the performance of other receivers is not impacted. The algorithm then determines the proximity to the "saving" threshold. The closer a receiver is to the threshold, the sooner it transmits its feedback. Any receiver below the "saving" threshold does not participate in the back-off because saving them will incur excessive retransmissions that violate the performance thresholds of "salvageable" ones. Other receivers above the threshold that hear r-DACK frames cancel their own per round. This approach ensures that the receiver with the most need is prioritized, improving network efficiency and reducing latency. Section IV-C details the design of r-DACK and its backoff equations.

IV. Design

We divide this section into the following: *i) Parameter Space Exploration*, describing the analysis of each rate control parameter (bandwidth, guard interval, spatial stream, coding rate, and modulation) from the traces collected that drive our design; *ii) Rate Selection*, describing the process of how our rate algorithm operates; and *iii) r-DACK*, an implicit coordination multicast feedback protocol that allows applications to provide their requirements (loss rate and goodput tolerance) so that consumers can request retransmissions without a worse-performing one dragging down everyone else.

A. Parameter Space Exploration

In this section, we thoroughly analyze how different parameter changes affect loss rates in various environments. We find that only a small fraction of parameter combinations are needed, and there is no need to explore all possible combinations for rate control. The findings below were consistent across data analysis in multiple variations: *i)* loss rate per data rate over various time granularity (a few minutes, per hour, 24 hours); *ii)* grouping data based on location condition (busy/empty, or LOS/NLOS) or based on distance. We provide

		Spatial Stream	1				2			
		Bandwidth	20MHz		40MHz		20MHz		40MHz	
		Guard Interval	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
MCS	Modulation	Coding rate	Data Rate (Mbps)							
0	BPSK	1/2	6.5	7.2	13.5	▨ 15.0	13	14.4	▨ 27.0	▨ 30
1	QPSK	1/2	13.0	14.4	27.0	30.0	26	28.9	54	▨ 60
2	QPSK	3/4	19.5	21.7	▨ 40.5	▨ 45.0	39	43.3	81	90
3	16-QAM	1/2	26.0	28.9	▨ 54.0	60.0	52	57.8	▨ 108	▨ 120
4	16-QAM	3/4	39.0	43.3	81.0	90.0	78	86.7	▨ 162	▨ 180
5	64-QAM	2/3	52.0	57.8	108.0	120.0	104	115.6	▨ 216	▨ 240
6	64-QAM	3/4	58.5	65.0	121.5	135.0	117	130.3	▨ 243	▨ 270
7	64-QAM	5/6	65.0	72.2	▨ 135.0	150.0	130	144.4	270	▨ 300
8	256-QAM	3/4	78.0	86.7	162.0	180.0	156	173.3	▨ 324	▨ 360
9	256-QAM	5/6	N/A	N/A	180.0	200.0	N/A	N/A	360	▨ 400

TABLE I: All Data rates used in the data collection. The shaded boxes are the candidate rates set.

insights into each parameter individually (modulation, spatial stream, guard interval, and bandwidth) while keeping in mind that the loss rate is impacted by the combination of parameters and not individual parameter change.

Data Collection Setup. We setup a radio dongle in V-MAC mode (ALFA AWUS036ACH 802.11ac) and start transmitting five back-to-back frames for each data rate. The experiment spanned 76 data rates with the parameter configurations shown in Table I. We setup four receivers per environment where each receiver is gradually further from the transmitter (typically one receiver in the same room LOS, a second outside the room across the transmitter, a third on another floor, and a fourth on another floor if possible and at the edge of the property) to see the impact of multipath and distance. We collected over two million packet traces in 12 environments (homes, offices, labs, and corporate), with each trace spanning 24 hours. We have deployed a medium analyzer [21] across the environments and found that there were sufficient variations in medium utilization (20%-80%) and traffic patterns. This ensures our observations are general across different environments. Data collection across more variations and with mobility can be conducted to validate the generalizability of our observations further.

1) No Monotonic Correlation between Data Rate and Loss: Higher data rates do not necessarily correlate with higher loss rates. This is because data rates in modern 802.11 standards change by five different parameters concurrently. Figure 1 shows two samples of 24 hours in two different environments (corporate, Figure 1a and home, Figure 1b). We can observe that only about 10-15 data rates along the dashed line are worth exploring, while the rest above the line have higher losses for the data rates they provide. This shows that some parameters increase the data rate without impacting the loss rate.

Based on the results displayed, there are multiple insights to note: *i)* data rate does not correlate monotonically with loss rate; as an example 180Mbps has a 10% loss rate while 78Mbps has a 50% loss rate (Figure 1a); *ii)* there is a small set of rates (i.e., those along the dashed lines) that have the lowest loss when providing a given rate, as we can see in our two samples. There are about nine such data rates in each environment.

2) Modulation: We find that modulation heavily affects the loss rate (which can result in 0-100%).

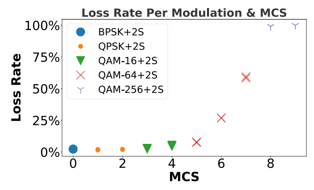
3) Spatial Stream: Spatial streams compound the loss rate when the modulation is incorrect. We find that in most environments between Modulation Coding Scheme (MCS) 6-9 loss rate can double and even go up to 90% while the single spatial stream's loss rate (in MCS 6-9) remains relatively low (<30%). However, we find that below MCS 6, two spatial streams have a lower loss rate than a single spatial stream. We believe this behavior occurs in lower MCS as 1SS has more noise (i.e., more transmissions), resulting in a higher loss rate.

4) Coding Rate: We find that with certain receivers when modulation and spatial streams are set correctly, the coding rate can impact the loss rate span by up to 70%. However, when the modulation and spatial stream are not correctly set, changing the coding rate does not impact the loss rate at all because, with an incorrect modulation and spatial stream, coding rate changes cannot save the frame.

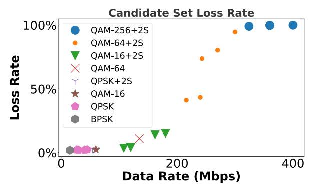
5) Guard Interval: Guard interval affects the loss rate slightly. We observe a maximum of 15% loss rate difference between short and long guard intervals across all loss rates.

6) Bandwidth: We find in most environments 40MHz performs better than 20MHz across all early MCS (0-5). We believe this is because such a higher bandwidth has less noise than 20MHz due to less traffic. However, when the modulation and spatial stream are too high for the environment, the loss rate can double when the bandwidth changes from 20 to 40MHz. However, another rate in 40MHz with a lower loss rate and higher data rate can be found; it is not worth considering any 20MHz rate.

7) Candidate Rate Set: We form a candidate rate set that can be used in all environments (i.e., those along dashed lines at the bottom, offering the lowest loss given a rate from all environments). It retains a monotonic increase with a loss rate. The process of selecting a candidate rate was the following: *i)* analyze the loss rate over variable time intervals (10 minutes, 30 minutes, hourly, 24 hours) for each rate across each environment; *ii)* find the data rates that have the least loss rate; *iii)* include other rates collected from other environments; *iv)* eliminate data rates that have minimal improvement (<0.5%



(a) Modulation



(b) Candidate set loss rate

Fig. 2: (a) shows loss rate difference among different modulations at two spatial streams and 40MHz. (b) shows monotonic loss rate increase of candidate set performance in a sample environment.

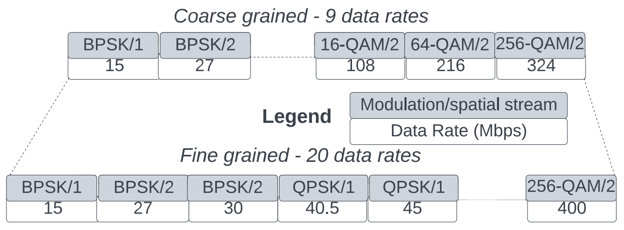


Fig. 3: Coarse and Fine search rates. The algorithm divides data rates into a coarse-grained set where changes to loss rate are high and less with fine-grained.

loss rate improvement) and major degradation in over 50% of traces (>15% loss rate degradation). We only eliminated 3 data rates, and their impact was on 90% of the traces. The candidate rates can be found in Table I in the shaded cells: 20 out of 76 combinations, a much smaller space to search. Figure 2b shows the candidate set loss rate in a sample environment. We observe a consistent correlation between data rate and loss increases among the candidate set. More analysis and evaluation about the candidate rate set can be found in Section V-A.

B. Rate Selection

Our rate selection process consists of *i) Exploration* that determines which rates to use and how to select them; and *ii) Performance Tuning*, detailing what kind of performance the application desires so that our system can provide it. The main issues the *Exploration* faces are the following: *i)* there are still twenty data rates to choose from, and that can require many frames to explore; *ii)* identifying when it's possible to increase the data rate without risking violating performance thresholds. *Performance Tuning* is needed to answer one question: how can the MAC optimize the tradeoff between loss rate and goodput while knowing the application's preferences?

1) Exploration: Our process operates on a round of frames that are within a stream. A round is a set of frames that are sent back-to-back before a feedback frame(s) from the receiver(s) comes providing information. There are two components: *Coarse and Fine-Grained Search*, defining how to search within the twenty candidate data rates, and *Stable Rate and Opportunistic Rate*, representing rates used within a round (and how many frames are sent at a rate). Each component addresses one of the two challenges described above.

Coarse and Fine Grained Search. We know that within the parameters, the modulation and spatial stream have the most significant impact on the loss rate; coding rate and guard interval affect the loss rate less. Therefore, finding the correct spatial stream and modulation combination before coding rate and guard interval can speed up the search. We divide our data rate search into *coarse* and *fine* as Figure 3 shows. We select rates based on modulation and spatial stream as our coarse tuning (in the first few initial rounds). Once the optimal modulation and spatial stream are found, the correct coding and guard interval are investigated through the next rounds. In the evaluation (Section V-C1), we show the benefits of such an approach, and in the discussion (Section VI), we elaborate on the need for the design in larger tables.

Stable and Opportunistic Rates. We divide the rates of any round between *i) the Stable rate*, the rate used for the majority of transmissions, and *ii) the Opportunistic rate* to experiment with a higher rate. The stable rate and opportunistic rates are re-evaluated after every round of feedback. The *Stable Rate* ensures meeting the stream's performance policy while the *Opportunistic Rate* attempts to explore higher data rates to improve the stream's performance without violating the performance policy. In Section V-C1, we show the benefits of our design and its impact on an application's performance. We set the initial ratio between stable and opportunistic rates to be equal (50% each rate, meaning half of the frames use the stable rate and others the opportunistic) as at initial transmission, we assume no knowledge of the optimal rate. After initial rounds, we set the stable rate to 90% and the opportunistic rate to 10% of the transmissions.

2) Performance Tuning: We designed a performance policy that enables applications to determine their requirements. To support edge applications where goodput and loss rate are required for successful transmission (e.g., live video transmission) and cases where loss rate is all that matters (e.g., latency tolerant file transfer), we form two policies: *i) loss rate* and *ii) loss rate and goodput* policies. Each of these policies allows upper layers (i.e., either transport, network, or application) to define their minimum loss rate and/or goodput needs. We have considered other forms of policies and could not find an application that needs it directly. We discuss this more in Section VI.

Initial Rates. We set the initial rates based on the application's needs. The application must have reasonable constraints for the system to work (elaboration on how to define reasonable constraints is in Section VI), we set the initial rates above the constraints by at least 10 Mbps to ensure meeting the goodput requirements. If the stream has only loss rate constraints, we set the initial rates on a conservative start (starting at the lowest candidate rates).

In/Decreasing Data Rate. The decision to move up or down the data rate ladder is tied to the current performance and the application's performance policy. The difference between the two policy types is the data rates available: the policy with goodput constraints prevents the algorithm from dropping below the goodput stated (e.g., if 20Mbps goodput is required, only nominal rates with goodput above 20Mbps can be selected). If the observed loss rate is above a certain threshold (e.g., $>$ 15% policies), the data rate is dropped in the coarse-grained level, otherwise in the fine-grained level. Increasing the data rate only occurs when the opportunistic rate demonstrates that it can meet the policy requirements. We choose a conservative increase in opportunistic data rate as the improvements are not necessary (since the algorithm starts with minimum requirements). This gives us the ability to gain goodput cautiously (without suffering sudden loss hikes).

With the algorithm and performance policy designed, we can now adjust the data rates to provide optimal performance based on a feedback representation from consumers.

C. r-DACK

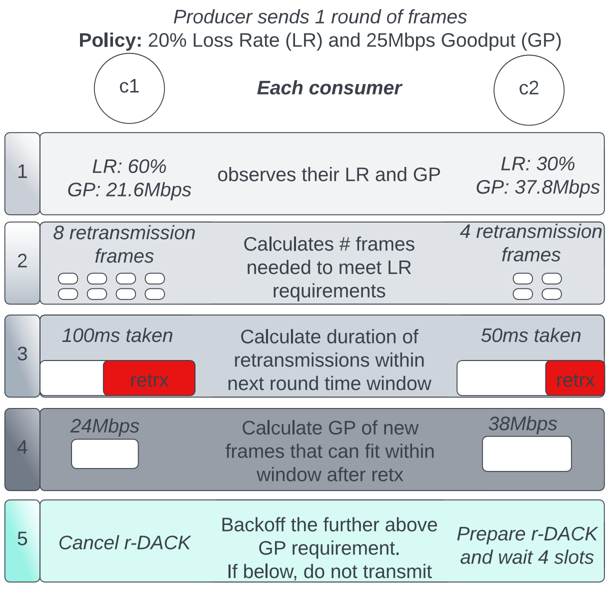
In this section, we design rate control DACK (r-DACK), which accounts for the goodput and loss rate policy. We extend prior work of the DACK feedback protocol [7] (described in Section II-A), where the receiver with the most loss reports first, without consideration of goodput constraints. We leverage the DACK protocol as is for the loss rate policy (since it allows the receiver with the most loss to report). We need to modify DACK so that each receiver estimates how much retransmission they need and estimates goodput cost to decide if they can be saved or not. r-DACK enables "salvageable" receivers with the most loss to report first.

r-DACK's main challenges are *i)* the need to implicitly coordinate among all consumers with no knowledge of their performances; *ii)* each consumer has to determine, based on limited observations, whether itself can be "salvaged" while meeting the performance policy (goodput) before participating.

1) Backoff Equation: A backoff equation is a calculation performed for the nodes to wait before sending a frame. There are two different backoff equations for the two policies. We leverage the original DACK backoff algorithm for the *Loss Rate Policy* explained in Section II-A. We account for retransmission failure by calculating the current loss rate based on the feedback and retransmitting enough frames. Even with the current loss rate, all receivers obtain enough frames to be below the constraint. Below, we describe our new backoff equation for the *Goodput and Loss Rate Policy*.

The goal of the feedback is to have the consumer that has the most number of frames missing yet is still "salvageable" after accounting for retransmissions (i.e., meeting required goodput) go first. We first explain the logic and workflow of what each consumer must do, then formulate it with equations.

In Figure 4 two consumers have received a round of data frames. They want to determine if they can get any missing frames retransmitted while still adhering to the policy's constraints in the next round. The process involves five steps that each consumer follows: *i)* observes its current loss rate and goodput, and it notes the stable and opportunistic rates the producer transmitted (which can be extracted from received frames); *ii)* calculates the number of frames needed in order to decrease the loss rate below the threshold. They also take into account the likelihood of retransmission failures



Producer sends 1 round of frames
Policy: 20% Loss Rate (LR) and 25Mbps Goodput (GP)
c1
Each consumer
c2
1
LR: 60%
GP: 21.6Mbps
observes their LR and GP
LR: 30%
GP: 37.8Mbps
2
8 retransmission frames
Calculates # frames needed to meet LR requirements
4 retransmission frames
3
100ms taken
retrx
Calculate duration of retransmissions within next round time window
50ms taken
retrx
4
24Mbps
Calculate GP of new frames that can fit within window after retx
38Mbps
5
Cancel r-DACK
Backoff the further above GP requirement.
If below, do not transmit
Prepare r-DACK and wait 4 slots

Fig. 4: r-DACK workflow, enabling the consumer with the most losses to save (without impacting goodput beyond constraints) to go first.

based on observed behavior; *iii)* estimates how much time the retransmissions will take within the time window of the next round; *iv)* based on the remaining time within next round's time window, considering the duration of the retransmissions, the consumer predicts the number of new data frames that the producer will be able to transmit and calculates the normalized goodput (i.e. after accounting for the impact of necessary retransmissions); *v)* if the normalized goodput is above the policy goodput, the consumer participates in backoff and feedback. The closer the normalized goodput is to the minimum goodput threshold, the less the consumer waits before sending their feedback. Below, we will explain each step in more detail and provide the necessary equations for the process.

At step 1, there are no calculations, but each consumer observes its loss rate (based on sequence numbers), goodput, and data rates (metadata from PHY) used during the last round. In step 2, the number of frames needed to meet policy requirements is calculated through the following equation:

$$n = (lr_{round} - lr_{policy}) \cdot n_{round} \quad (1)$$

Where n denotes the number of frames needed to be below the loss rate threshold; lr_{round} denotes the loss rate of the round, and lr_{policy} denotes policy's loss rate requirements, multiplied by a constant number of frames in a round denoted as n_{round} (set empirically in Section IV-A). In step 3, we estimate the duration of the retransmission needed to get the consumer below the policy's loss rate threshold. We account for the potential loss of retransmission frames as well. We do so using the following equation:

$$\tau = \frac{n}{1-\ell}. \quad (2)$$

τ denotes the number of frames needed to transmit while accounting for the new frames' retransmission failure. n is the number of frames needed to be received as calculated in (1), and ℓ denotes the predicted loss of future transmissions (i.e., how many frames retransmitted may be lost). We set ℓ to be the policy's loss rate goal (i.e., if the policy's loss rate threshold is 10%, we assume only a 10% loss rate of retransmissions). We choose to be optimistic with ℓ and aim for the lowest loss rate value of the policy, assuming that retransmission will bring the loss above the threshold. It is better to assume a lower number of frames needed and be incorrect. Overestimating the number of missing frames may lead to salvageable consumers giving up. Further discussion on how to improve ℓ is in Section VI. Afterwards, we use the following equation to calculate the total duration of retransmissions:

$$T_{retx} = \tau \cdot \left(\frac{size_{frame}}{rate_{retx}} + preamble \right) \tag{3}$$

$rate_{retx}$ (measured in bits per μs (bpμs)) denotes the data rate selected for retransmission (known through the stable rate and the algorithm). $size_{frame}$ denotes the payload size of a single frame in bits. τ denotes the number of frames needed to retransmit for the consumer to meet the policy threshold calculated in (2). We also add the preamble duration (96μs) of however many frames need to be transmitted. After obtaining the duration cost of retransmission, we subtract it from the total window duration, which is used to calculate the goodput.

The total window duration (microseconds) is calculated by:

$$T_{window} = n_{round} \cdot \left(\frac{size_{frame}}{gp_{policy}} + preamble \right) \tag{4}$$

gp_{policy} denotes policy's goodput (bpμs), while other variables have been explained in (3). This equation enables us to know the minimum required duration of transmission for a round of frames to retain the policy's goodput.

In step 4, we calculate the adjusted speed based on the knowledge of how many frames can be transmitted within T_{retx} obtained from the previous equations. We assume the frame size is similar to previous frames on the same stream and know that the new data rate is either the stable rate or one rate lower/higher. We can calculate the duration of one new frame transmission as follows:

$$T_{nf} = \frac{size_{frame}}{rate_{sn}} \tag{5}$$

$rate_{sn}$ denotes the new frame's transmission rate (obtained by feeding the algorithm current transmission rates and r-DACK feedback). Then, we can estimate how many new frames will fit in the time left in the window using the following equation:

$$n_{new} = \left\lfloor \frac{T_{window} - T_{retx}}{T_{nf}} \right\rfloor \tag{6}$$

Afterwards, we can calculate the new goodput of new frames:

$$gp_{adj} = n_{new} \cdot \frac{size_{frame}}{T_{window}} \tag{7}$$

n_{new} denotes the number of new frames that can be transmitted at the next round time window. We are now only missing determining how close the goodput is to the minimum bound. It is calculated through the following equation:

$$gp_{norm} = \frac{gp_{adj} - gp_{min}}{gp_{max} - gp_{min}} \tag{8}$$

gp_{norm} indicates the normalized goodput. If the goodput is negative, that means to meet the consumer needs of loss rate, the goodput will drop below threshold requirements, and the consumer does not provide feedback; gp_{max} denotes the maximum goodput possible. It is obtained based on calculating the stable rate goodput from the previous round (we account for the frame size, its preamble, and the contention window duration) and assumes a 100% reception rate. gp_{min} denotes the minimum goodput required, defined in the performance policy. gp_{adj} denotes the estimated goodput in the next round, after taking off the cost of retransmissions that the consumer needs (calculated in step 4 and equation 7). This allows us to identify the consumer with the most missing frames where the adjusted speed is closest to the minimum. For example, to drop the consumer's loss rate from 45% to 25%, the goodput of the next round will be 20 Mbps instead of 40Mbps, and the minimum bound is 19%. Thus, the ratio is 0.05.

Finally, the backoff equation can be represented as:

$$T = \alpha \cdot gp_{norm} \cdot \sigma \tag{9}$$

α denotes the slot duration between two data frames of the same stream (excluding frame time), indicating medium contention and how often radio can transmit; σ is a constant that distributes the goodput result over the slots evenly, bounding the wait time of r-DACKs by the number of slots (e.g., if σ is 15 then there are 15 different slots to choose from). Through the equations above, a consumer is able to estimate if they can be saved without violating the goodput constraints or not, and if saved, how long to wait based on need.

2) Design Decisions: We choose to have a producer start by retransmitting the oldest frames first (based on sequence number), then keep moving forward until we reach the limit of retransmissions for a round. We choose the oldest frames first because the latter frames have more opportunities to be retransmitted through future rounds of r-DACK.

We set the round size to 20 frames for the following reasons: *i)* r-DACK is time-consuming due to multiple calculations; *ii)* it is best to get feedback on large sets of frames to eliminate significant, abrupt changes in losses in a small window.

We set σ to 15 for the following reasons: *i)* it is a wide distribution to prevent consumers transmitting at the same slot and enough space among each transmission to hear each other and decide before sending their own and hear each other to cancel; *ii)* it is short enough so that the r-DACK arrives at the producer before the end of the next round. We allow consumers to cancel their r-DACKs after hearing two others of the same round.

We find that our system is able to receive r-DACKs and adjust one round in advance as desired. We also find that consumers that are the most in need can be saved without violating goodput policy. Detailed results can be found in Section V-C1.

V. Evaluation

We implement our system on an 802.11ac NIC through a commodity USB WiFi Interface (ALFA AWUS036ACH). We run the system on Raspberry Pi 4 Kernel 5. We modify the kernel driver and reverse engineer monitor mode to support transmitting V-MAC frames accordingly. We start from monitor mode since it does not transmit beacon frames and allows overhearing all the frames. We add the ability to transmit different data rates on a per-frame basis through hooks from the firmware to the userspace. The main challenge is the ability to alternate all different parameters of data rate per frame transmission. The other challenge we faced was increasing the size of the frame to larger than the standard frame size (1500 bytes) without using AMPDU. AMPDU expects a BACK (Block Acknowledgment) and a station in the firmware, which is infeasible in multicast.

Through a series of modifications to the memory structures passed to the kernel, we are able to access up to 40MHz bandwidth, two spatial streams, all the modulations, guard intervals, and coding rates. We were unable to reverse engineer 80MHz reliably as we did with 40MHz to perform experiments. However, we believe our design can scale without change to the whole space of parameters, should more accessible hardware become available. We believe the set of candidate data rates will need re-evaluation to find which new rates will remain in the large rates table.

Different NIC. We have repeated most of our experiments on a different 802.11n NIC (AR9271) to ensure the correctness and validity of the results. We did not find any of our observations changing besides performance and rates since it is another standard.

A. Impact of Candidate Rate Set

Setup. We setup a testbed of 5 receivers in Line Of Sight (LOS) to each other and a transmitter. We determine the optimal data rate through trial and error (where goodput is maximized). We perform these experiments in a place with no other radios or anyone and validate via Wireshark on background traffic. We setup the experiment by having consumers request one stream of 900 data frames by sending an Interest every 100 data frames. We have the producer artificially waiting 10 seconds for the very first Interest for us to start the application across all consumers manually. We then let our algorithm operate with and without the candidate rate set to observe the difference.

2.5× convergence improvement. We find that our algorithm reaches the optimal data rate in a maximum of 2-3 rounds of feedback, while without the candidate data rate set, it takes 6-9 rounds. We have also noticed that without a candidate data rate set, the algorithm can get stuck at a local maxima because the next data rate has a higher loss rate while the one after has much less than the current rate. (e.g., 65Mbps has 70% loss rate while 135Mbps has 30%). Further, regardless of the selection process or data rate change, if the data rates are not organized based on proper loss rate with "unusable" data rates taken off, a case where the algorithm gets stuck in a local maxima or a sub-optimal data rate is inevitable.

Similar results have been shown that higher data rates can result in lower loss rates in legacy standards [5], [22]. Our work is the first of its kind where multiple environments are tested, and a dataset is defined using 802.11ac. We have also found clear justifications as to why some data rates are performing better across modulation, coding rates, spatial streams, and bandwidths. These results confirm the benefits of having candidate data rates set and show that organizing data rates according to loss rate while bearing in mind data rate is critical for proper rate control performance.

B. Coarse and Fine-Grained Search

We retain the setup mentioned in our previous experiment to test the impact of coarse and fine-grained search (Section V-A). We observe that it takes two rounds of feedback on average to converge to the ideal data rate. Without a coarse-grained search, we find that it can take between 4-7 rounds of feedback, and this is due to the larger map to search in. These 4-7 rounds cost 80-140 data frames (which can consume up to 0.2 seconds assuming transmission at 6Mbps with 1500 bytes frames). This demonstrates that the coarse and fine-grained search converges on the optimal data rate 2-3X faster than using the entire table. We further discuss the benefits of such an approach on larger space tables in Section VI.

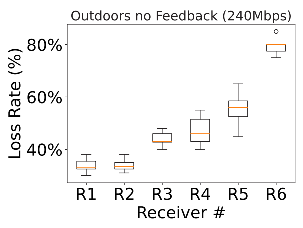
C. r-DACK

With the impact of candidate rates set and coarse-then-fine-grained search evaluated, we evaluate r-DACK in various environments under different constraints below while leveraging both design components.

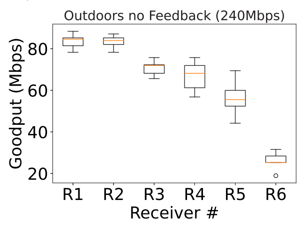
We start by performing outdoor experiments. We ensure there is no background traffic by using Wireshark and monitoring the channels we intend to use. We then proceed to indoor environments (home and corporate) and compare against: no feedback mechanism (i.e., just broadcast), r-DACK loss rate only policy, and two r-DACK policies with different goodput and loss rate thresholds.

We setup the experiment (similar to previous experiments) by having consumers request one stream of 900 data frames by sending an Interest every 100 data frames. We have the producer artificially waiting 10 seconds for the very first Interest for us to start the application across all consumers manually. Afterward, the producer transmits right after receiving any interest, with all consumers subscribed. We also experimented with an interest every 20 frames instead of 100 but saw little impact on performance besides increased latency due to extra interest traffic. Such changes to latency are orthogonal to multicast protocol performance.

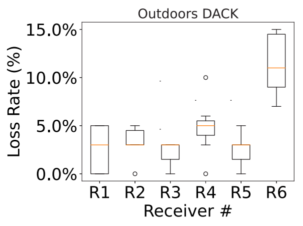
1) Outdoors: Our experiments without interference show how our protocol operates in detail, and experiments with interference show the system's robustness. For interference, we place 2 Raspberry Pis outdoors and have each send random-sized frames from 300 bytes to 2000 bytes every 5-10 ms.



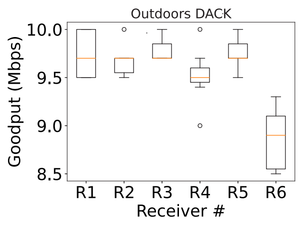
(a) Loss Rate (240Mbps)



(b) Goodput (240Mbps)

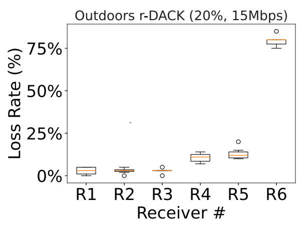


(c) r-DACK (10%) Loss Rate

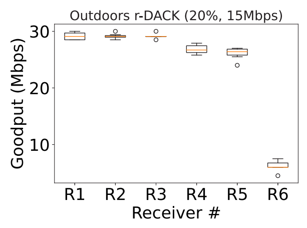


(d) r-DACK (10%) Goodput

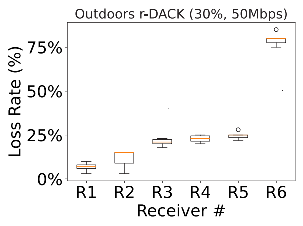
Fig. 5: (a) and (b) show broadcast performance. (c) and (d) show r-DACK with a 15% loss rate policy. Receivers are ordered from closest to furthest.



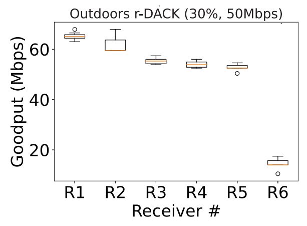
(a) r-DACK Loss Rate



(b) r-DACK Goodput



(c) r-DACK Loss Rate



(d) r-DACK Goodput

Fig. 6: (a) and (b) show results of r-DACK with performance policy (20%, 15Mbps); (c) and (d) show results of r-DACK with performance policy (30%, 50Mbps).

Without Interference. Figures 5a and 5b show results of broadcast at 240Mbps data rate to establish a baseline. We observe loss rate fluctuations across receivers (Figure 5a) ranging from 25% to 80%, leading to goodput ranges (Figure 5b) of 20-75Mbps. We can observe high fluctuations in both goodput and loss rate, which makes it very difficult to run applications such as video streaming. Further, if a higher data rate is selected, the loss rate goes up to almost 100%, and if the lower rate is selected, the loss rate still fluctuates up to 80% with a reduction in goodput.

When we apply r-DACK with a loss rate policy of 15%, we can observe loss rate reduction (Figure 5c) and goodput ranges (Figure 5d) of 8.5-10Mbps. This demonstrates the ability of the r-DACK loss rate policy to reduce the loss rate to desired levels. However, this comes with the cost of retransmissions to satisfy all consumers, including R6, which had nearly 80% loss rate; reducing R6 to 15% loss rate led to 4-10X reduction in goodput for all receivers.

Next, we apply two different r-DACK policies, one with 20% loss rate and 15Mbps goodput and another with 30% loss rate and 50Mbps goodput. Figures 6a and 6b show loss rate and goodput for the (20%, 15Mbps) policy. We observe that the algorithm correctly gave up on R6 (since R6 cannot meet these targets) and met the policy requirements across other receivers. Similarly, with Figures 6c and 6d, we can again see R6 is ignored, but the rest of the receivers have met both the loss rate (30%) and goodput (50Mbps) requirements. This shows that r-DACK can balance between goodput and loss, giving up only the worst receiver while still saving the rest.

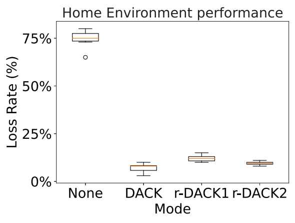
Rate oscillation. It is common with rate control designs to see the producer's rate going up and down every few frames, exhibiting uncertainty of which rate is optimal. We do not observe rate oscillation behavior with our system. The reason our system is not as susceptible to rate oscillation is that our system takes a round of frames (20 frames) before making any decision, leading to more robustness against rate oscillation.

Impact of distance and multipath. We test our system's performance by placing the receivers at different distances and behind vehicles from the transmitter. We observe that our system lowers the optimal rate while meeting the policy thresholds and saving all salvageable receivers. We have tested 30% loss rate with 40Mbps, which was met by 5 receivers out of 6, and 20% loss rate with 20Mbps, which was met by all six receivers. This demonstrates that our system ability to handle different policy constraints under NLOS scenarios.

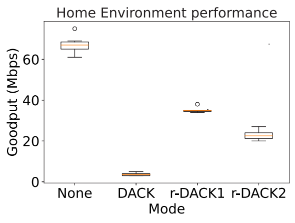
2) Indoors: We tested our system in multiple homes and confirmed the ability of r-DACK to meet different policy requirements. We rotate among four different modes to demonstrate the different performances and do ten runs. The four modes are one without any feedback to get a baseline of environment and system performance in the environment, second with DACK where the worst consumer can drag goodput performance, and two other modes with different r-DACK policies demonstrating the flexibility in tuning performance.

Figure 7 shows the results of the four modes None (i.e., no feedback at 240Mbps), DACK (10% loss rate policy), r-DACK1 (30% loss rate and 40Mbps), r-DACK2 (20% loss rate and 20Mbps). Our results show that r-DACK was able to retain the required loss rate results (Figure 7a) in the three different policies specified. In r-DACK1 and r-DACK2, it was also able to retain the goodput requirements: 40Mbps and 20Mbps, as shown in Figure 7b. These results show the robustness and resilience of our system in a noisy environment.

Increasing loss rate policy by 3% increments and goodput. We test our system's flexibility in an indoor environment with five consumers and one producer where we start from 10% loss rate to 40% loss rate in 3% increments, with required goodput of 45Mbps. We observe the system's



(a) Loss Rate



(b) Goodput

Fig. 7: We can observe that r-DACK was able to retain its performance policy across the three modes and not impact goodput relative to None (i.e., broadcast).

ability to maintain the goodput and abandon the consumers that could not retain certain loss rates. This demonstrates the system's flexibility and ability to fine-tune loss rate performance requirements successfully.

Corporate office and lab. Our results in two corporate offices and two research labs yielded similar results to home environments (four modes, ten runs per mode), which demonstrate the system's robustness across environments. Performing our experiments in three different types of environments (corporate, lab, home) and multiple places per environment demonstrates: *i)* the system's resilience and ability to adapt; *ii)* the system's robustness in meeting performance requirements and 'salvaging' those receivers that can be saved; *iii)* we can, within reason, generalize the results to other untested environments.

D. Scalability

We have tried experiments with up to 12 receivers and observed a cap to the overhead increase as the number of receivers goes beyond 7-8. The cap occurs when extra receivers do not transmit r-DACK and cancel their own, satisfied with others' requests. This does not ensure that the cap will remain constant for the following reasons: *i)* consumers may not be able to hear each other, and thus, the cancellation policy may not trigger; *ii)* multiple consumers end up in the same slot and try sending at the same time, leading to the inability to cancel the timer. However, this does not result in frames colliding due to CSMA operating. This shows that our system operates appropriately and can prevent redundant overhead by eliminating numerous r-DACKs.

Projection analysis. Given our hardware implementation limitation, we are capped by 100 Mbps goodput due to its inability to use A-MPDU in multicast. We analyze how much overhead we have with our data rate and frame size, then scale accordingly to see how high our multicast rate goodput can go up. Given how our algorithms performed at 100 Mbps with limited hardware, we can scale up with adjusted frame size when we have access to A-MPDU and the entire table to reach up to 1 Gbps goodput while retaining the policy's threshold (e.g., 20% loss rate and 950 Mbps), assuming that the environment will allow for such data rate to work.

Design limitations. Our design currently makes a few assumptions that will need to be revisited based on real-world application needs: *i)* We choose to retransmit the oldest frame first to provide every frame with the highest chance to be received. However, this may not be the best policy for some applications that may prefer the latest (e.g., multi-vehicle trajectory planner with missing data compensation); *ii)* the design assumes large amounts of data (100+ data packets in a stream) to be able to perform well. There can be multiple small sets of data where each has its stream but shares the same set of receivers beneath. A protocol that enables sharing optimal rates across streams is necessary for the system to perform well in such a scenario. Both scenarios' limitations can be worked around with extended research and engineering work given the current design's architectural flexibility and approach; we leave such extensions for future work.

VI. Discussion

Data rate constraints based on frame size. Our analysis and studies show that each frame rate should have a frame size lower bound and an upper bound. The lower bound is due to the fixed preamble duration (96μs), which can consume most of the air time for small frame sizes transmitted at high speeds (e.g., a 1400 byte payload frame transmitted at 400Mbps consumes 28μs which is about 3.5X less than the preamble). Besides preamble duration, accounting for contention window duration and other overhead is critical to obtain true goodput measurements. The upper bound stems from the limited duration of channel coherence time (e.g., 5ms) in which frames have to be completely transmitted. We have observed that transmitting a large frame beyond the coherence time led to failure. Therefore, for each data rate, a lower bound (to prevent transmission overhead from consuming most of the transmission) and an upper bound (to be within channel coherence) based on frame size is important. This matters to r-DACK as rate control needs to account for frame sizes while choosing the data rate for transmission. These constraints are to be placed on top of the design so that each frame being transmitted has a small set of data rates to select from within the candidate rates set. We leave such work and exploration for future work.

Coarse and fine-grained search on larger spaces. Our evaluation of our search technique is currently limited due to the current hardware capabilities. We believe our coarse and fine-grained search algorithm will show higher impact results in larger data rate tables (e.g., in full 802.11ac with all bands available, the table grows from 76 data rates to 232 data rates, which can result in a reduction to 58 rates).

Choosing the right performance requirements. Our system relies heavily on applications choosing the right goodput and loss rate requirements for themselves. If incorrect or too aggressive constraints are selected, the consumers may give up as they will not be able to meet the requirements. Impossible policy requirements depend on multiple aspects: *i)* communication condition between the consumer(s) and producer; *ii)* the optimal data rate for transmission; *iii)* the size of the frames transmitted. Thus, we encourage applications to choose the minimum requirements for the application to operate and let the algorithm improve goodput slowly while retaining the required loss rate.

Mobility Impact. We have not tested our system in mobile or partially mobile (i.e., either transmitter or receiver moving) environments. Based on previous works [15], [23], channel coherence is significantly impacted by mobility (e.g., 648μs at 5GHz with 100 km/hr), affecting transmission duration. It also changes the channel standard properties that are assumed during stationary or low mobility (1-3mph) environments. Thus, the process of rate selection and tolerances to change rate (up/down) will need to adapt. Existing works [15] have made both approaches and shown promising results. We believe the same can be added to our design in the future if needed for mobility. We leave full design for mobility as future work.

Chipset Variation and Performance. Other NICs can lead to different results due to the following reasons: *i)* cross-stack communication speed can differ, causing different delays; *ii)* medium sensing and antenna sensitivity vary per NIC, thus causing missing slots or sending when the medium is busy. We have tested two NICs (AR9271 and RTL8812AU) and observed the same behavior we detail in Section III-A, confirming the phenomenon is not NIC-related. However, there were different performance results, which were caused by different frequency ranges, rate tables, and antenna numbers. Further experiments to verify across NICs are needed.

Experimental Work Comparison. As mentioned in Section II-C, prior approaches designate a leader [18] or form a group [20] and then provide feedback. Such designs have setup overhead prior to communication, and the overhead increases linearly with each new multicast group. Meanwhile, our system is group-less and does not require a prior agreed-upon elected receiver to provide feedback nor any setup overhead, offering true peer-based communication and better support for medium-changing wireless environments. An analysis of the performance of both designs with scenario tradeoffs can be made in the future.

VII. Conclusion

In this paper, we introduced an application of adaptive multicast rate control with thorough parameter analysis, the first of its kind. We perform extensive data collection, analyzing each data rate's efficiency across multiple receivers across three environments (corporate, lab, and home). Through our analysis, we were able to reduce data rate choices by 3.8×. We also introduce r-DACK, a rate-adaptive multicast feedback design that enables applications to provide their desired performance constraints (loss rate, goodput, and loss rate). Our prototype implementation on 802.11ac radios shows five consumers obtaining 50Mbps goodput with a 30% loss rate.

References

[1] Lingyang Song, Zhu Han, Zhongshan Zhang, and Bingli Jiao. Non-cooperative feedback-rate control game for channel state information in wireless networks. *IEEE Journal on Selected Areas in Communications*, 30(1):188–197, 2011.

[2] Ioannis Pefkianakis, Starsky HY Wong, Hao Yang, Suk-Bok Lee, and Songwu Lu. Toward history-aware robust 802.11 rate adaptation. *IEEE Transactions on mobile computing*, 12(3):502–515, 2012.

[3] Varun Gupta, Craig Gutterman, Yigal Bejerano, and Gil Zussman. Experimental evaluation of large scale WiFi multicast rate control. *IEEE Transactions on Wireless Communications*, 17(4):2319–2332, 2018.

[4] Fan Wu, Wang Yang, Ju Ren, Feng Lyu, Peng Yang, Yaoxue Zhang, and Xuemin Shen. NDN-MMRA: Multi-stage multicast rate adaptation in named data networking wlan. *IEEE Transactions on Multimedia*, 23:3250–3263, 2020.

[5] David Murray, Terry Koziniec, Michael Dixon, and Kevin Lee. Measuring the reliability of 802.11 WiFi networks. In *2015 Internet Technologies and Applications (ITA)*, pages 233–238. IEEE, 2015.

[6] Chi-Yu Li, Syuan-Cheng Chen, Chien-Ting Kuo, and Chui-Hao Chiu. Practical machine learning-based rate adaptation solution for Wi-Fi NICs: IEEE 802.11 ac as a case study. *IEEE Transactions on Vehicular Technology*, 69(9):10264–10277, 2020.

[7] Mohammed Elbadry, Fan Ye, Peter Milder, and Yuanyuan Yang. Pub/Sub in the Air: A Novel Data-centric Radio Supporting Robust Multicast in Edge Environments. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 257–270. IEEE, 2020.

[8] Dimitris Vassis, George Kormentzas, Angelos Rouskas, and Ilias Maglogiannis. The IEEE 802.11g standard for high data rate WLANs. *IEEE Network*, 19(3):21–26, 2005.

[9] Eldad Perahia. IEEE 802.11n development: history, process, and technology. *IEEE Communications Magazine*, 46(7):48–55, 2008.

[10] Boris Ginzburg and Alex Kesselman. Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n. In *2007 IEEE Sarnoff Symposium*, pages 1–5. IEEE, 2007.

[11] Matthew S Gast. *802.11ac: a survival guide: Wi-Fi at gigabit and beyond*. O'Reilly Media, Inc., 2013.

[12] Wei Yin, Peizhao Hu, Jadwiga Indulska, and Konstanty Bialkowski. Performance of mac80211 rate control mechanisms. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 427–436, 2011.

[13] Wei Yin, Peizhao Hu, Jadwiga Indulska, Marius Portmann, and Ying Mao. MAC-layer rate control for 802.11 networks: A survey. *Wireless Networks*, 26:3793–3830, 2020.

[14] Ioannis Pefkianakis, Suk-Bok Lee, and Songwu Lu. Towards MIMO-aware 802.11n rate adaptation. *IEEE/ACM Transactions on Networking*, 21(3):692–705, 2012.

[15] Seongho Byeon, Kangjin Yoon, Changmok Yang, and Sunghyun Choi. STRALE: Mobility-aware PHY rate and frame aggregation length adaptation in WLANs. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[16] Jong-Seok Kim, Seong-Kwan Kim, and Sung-Hyun Choi. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. *The Journal of Korean Institute of Communications and Information Sciences*, 31(2A):154–167, 2006.

[17] Syuan-Cheng Chen, Chi-Yu Li, and Chui-Hao Chiu. An experience driven design for IEEE 802.11ac rate adaptation based on reinforcement learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[18] Nakjung Choi, Yongho Seok, Taekyoung Kwon, and Yanghee Choi. Leader-based multicast service in IEEE 802.11v networks. In *2010 7th IEEE Consumer Communications and Networking Conference*, pages 1–5. IEEE, 2010.

[19] Wan-Seon Lim, Dong-Wook Kim, and Young-Joo Suh. Design of efficient multicast protocol for IEEE 802.11n WLANs and cross-layer optimization for scalable video streaming. *IEEE Transactions on Mobile Computing*, 11(5):780–792, 2011.

[20] Xinbing Wang, Luoyi Fu, and Chenhui Hu. Multicast performance with hierarchical cooperation. *IEEE/ACM Transactions on Networking*, 20(3):917–930, 2011.

[21] Mohammed Elbadry, Fan Ye, and Peter Milder. Aletheia: A lightweight tool for WiFi medium analysis on the edge. In *ICC 2021-IEEE International Conference on Communications*, pages 1–7. IEEE, 2021.

[22] Giuseppe Bianchi, Fabrizio Formisano, and Domenico Giustiniano. 802.11b/g link level measurements for an outdoor wireless campus network. In *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, pages 6–pp. IEEE, 2006.

[23] Yaxiong Xie, Zhenjiang Li, and Mo Li. Precise power delay profiling with commodity WiFi. In *Proceedings of the 21st Annual international conference on Mobile Computing and Networking*, pages 53–64, 2015.