This article has been accepted for publication in IEEE Communications Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2023.3269526

Proxy Path Scheduling and Erasure Reconstruction for Low Delay mmWave Communication

David A. Hayes, *Senior Member, IEEE,* David Ros, Øyvind Ytrehus

***Abstract*—In 5G and future cellular systems, use of multiple millimeter-wave (mmWave) links in parallel can enable interactive applications that demand consistent low delays with very high data rates. In previous work we introduced a mmWave multipath proxy that preemptively manages a set of mmWave paths to guarantee a steady data rate with high confidence. Here we explore the impact choice of packet sending path has on packet delay in this dynamic system, as well as the potential benefits *erasure reconstruction codes* (ERC) may provide in mitigating packet reordering delays at the receiver. We find that capacity aware scheduling improves packet delay, but that improvements diminish relative to the staleness of the capacity knowledge. ERC almost completely eliminates head-of-line blocking delays at the receiver, but any benefit is diminished by additional queueing delays induced by parity packets, unless they are selectively sent according to available capacity.**

***Index Terms*—Multipath scheduling, mmWave, erasure reconstruction.**

I. Introduction and Problem Statement

MILLIMETER-WAVE (mmWave) radio links allow data rates of several Gbps to be achieved in 5G networks, and they will remain a key element in 6G and future wireless systems. The downsides of mmWave are the need for line-of-sight (LoS) propagation, as well as the potentially severe impact of atmospheric conditions on radio propagation. For instance, data rates in non-line-of sight (NLoS) conditions can drop as low as <1% of typical LoS rates.

In spite of these shortcomings, mmWave radio is touted as an enabler of novel use cases for cellular networks, like advanced emergency communications or extended reality, based on ultra-high-definition interactive video. However, for these applications to work properly the network needs to ensure that bit rates are not only very high, but also stable and consistent, together with stable low latency. Vu *et al.* [1] look at this in the context of multi-hop paths in self-backhauled mmWave networks, and Dogan *et al.* [2] in terms of resilient paths through multi-connected multi-hop mmWave networks. We investigate this from the perspective of a multi-path proxy directing packets over a changing number of possible mmWave links to a single receiver, to achieve steady high rates and low delays.

In [3, 4], we designed a *mmWave multipath proxy* and evaluated it through packet-level simulations. Fig. 1 shows the system architecture and Fig. 2 provides an abstract representation of the multipath mmWave system under study. The proxy queues incoming packets and schedules each packet for transmission over one among a set of mmWave paths. Based on current network conditions and a Markov-modulated fluid queue (MMFQ) model, the proxy uses a *short-term prediction* of the future state of the queue (i.e., over a time horizon of a couple of hundred ms) to periodically adjust the set P of paths used for transmission. That is, the proxy dynamically adds/removes paths to/from P, in order to satisfy the application's quality-of-service (QoS) constraints with the lowest number of paths. Our focus is *not* on "greedy" applications that just want to maximize throughput (like many TCP-based applications), but rather applications that demand both consistent, high data rates *and* low delays. A typical scenario would be that of a mobile real-time application like immersive 3D video or ultra-high definition extended reality.

Our work in [3, 4] focused only on the *path management* element, using the aggregate capacity, C, to send packets for simplicity. This allowed us to show the feasibility and benefits

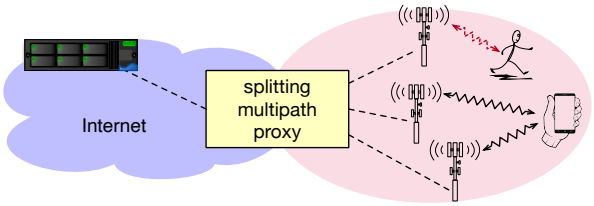


Fig. 1. System architecture overview. MmWave base stations are connected to a *splitting multipath proxy*, located in close network proximity to the base stations (three depicted in this example, for clarity, with one radio path being blocked by a pedestrian). The proxy gathers capacity information about the set of mmWave paths at its disposal to schedule packets over paths.

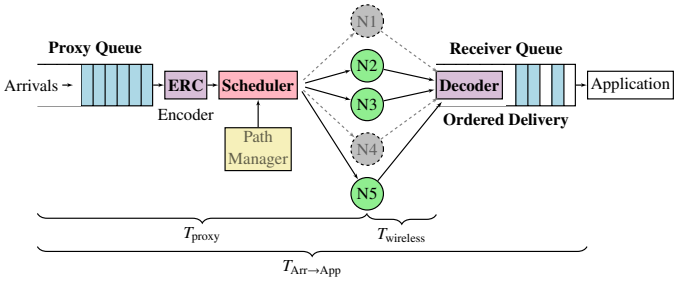


Fig. 2. The multipath proxy dynamically manages a set of mmWave paths (5 in this figure, labeled N1-N5). The proxy's *path manager* periodically (every 150 ms here) updates the set of paths to be used for packet transmission (shown as solid green circles), based on a probabilistic prediction of the proxy queue state over a short time horizon of 200 ms. Paths are chosen so that the aggregate capacity satisfies the application's requirements with confidence using the CDF-based path manager introduced in [3, 4]. In this paper we focus on the packet *scheduler*'s choice of paths to send packets and its interaction with an optional *erasure-reconstruction coder* (ERC), which can be used to protect against packet reordering delays at the receiver. T_{proxy}, T_{wireless} and $T_{\text{Arr}\to\text{App}}$ denote packet delays respectively: inside the proxy, over the wireless link, and total delay from arrival at the proxy to delivery to the application.

Manuscript received 4 December 2022, etc.

David Hayes did this work while at SimulaMet, Oslo, Norway (email: david.hayes@ieee.org); David Ros is with Simula Research Laboratory, Oslo, Norway (email: dros@simula.no); Øyvind Ytrehus is with Simula UiB and the University of Bergen, both in Bergen, Norway (email: oyvindy@simula.no).

of a predictive control for path management purposes. Though the path manager tries to ensure that C will remain large enough to carry the application traffic over the control interval (150 ms here), it does not select *which* path to send a packet on. As discussed in [4], the scheduling technique may have a strong impact on performance since paths may have very different bit rates in a given instant. Considering a simple Round Robin (RR) scheduler just cycling to the next path irrespective of capacity, it may choose a NLoS path even when a LoS path is available for use. If the LoS path has 100× the capacity of the NLoS path, the unnecessary reordering will cause head-of-line (HoL) blocking at the receiver for the 100 packets sent on the LoS path which now need to wait for the packet sent over the much slower NLoS path. More sophisticated scheduling methods can prioritize higher-capacity over lower-capacity paths, to minimize the chances of reordering. We study two such algorithms in section II, illustrating their impact on the total packet delay and the improvement they offer over a naive scheduler like RR.

No scheduling algorithm can guarantee that a packet will not be sent over a very low capacity path, whether due to the lack of available high capacity paths or to stale knowledge of path capacities (discussed later in section IV). *Erasure-Reconstruction Codes* (ERC) [5] can be used to mitigate the effect of packet delays on a slow path, and thus lower the probability of HoL blocking. This is done at a cost since more transmission capacity is needed to carry the parity packets. We evaluate the potential benefit of ERC and its relation with scheduling in section III.

The results in sections II and III rely on the assumption that the path manager and the scheduler have perfect knowledge of the capacity of all available paths when they make a path selection decision. In practice, however, information about changes in path capacities may not be known until after a delay. Therefore, in section IV we further explore the impact of imperfect (i.e., delayed) knowledge on path scheduling decisions, with and without ERC.

II. Path Scheduling for Dynamic Changing Paths

We use the term *path* to refer to one of the possible routes that a packet may take from the proxy to the mmWave receiver over a particular mmWave link. We assume that the proxy is able to route each individual packet over a particular path, that the capacity of a path is only limited by the mmWave link, and that the transit times from the proxy to the base stations are insignificant (refer to Fig. 1 and 2). As in [4], the mmWave link is modelled using the 3GPP 5G urban street canyon channel model, UMi [6].

In this paper we assume that all packets are of equal value to the receiving application and limit our consideration to a simple FIFO queueing discipline for the proxy queue. More advanced queueing disciplines may be beneficial, especially where some packets are more valuable than others, and will be examined in future work. The capacity of mmWave channels fluctuates due to shadow fading, but drops dramatically when LoS communication is temporarily blocked (NLoS). These drastic capacity changes mean the choice of which of the currently available paths to send a packet on can significantly impact its transit time to the receiver, which in turn can cause head-of-line blocking delays at the receiver. Here we investigate that effect with three simple schedulers[1]: Round Robin (RR), Highest Capacity First (HCF) and Enhanced Highest Capacity First (HCF+). We do *not* use erasure-reconstruction coding in these initial experiments to better isolate the effect of the scheduling discipline.

Both RR and HCF are work conserving schedulers. RR simply chooses f_1 to send a packet[2], and paths that become free are placed at the end of F. HCF chooses the free path with the highest capacity, $f_{\max}$. HCF+ is a non-work conserving heuristic enhancement of HCF that actively avoids sending packets on low capacity paths when there is sufficient total capacity. In these experiments, we define a low capacity path as having a capacity <10%[3] of the application's required sending rate, A. With HCF+, a packet will be sent over $f_{\max}$ only if $(C^{(f_{\max})} > 0.1A \vee C < A)$, else the scheduler will wait until another path with capacity $> 0.1A$ becomes available.

We use the simulator described in [4] with the predictive CDF-based path manager. Table II describes the relevant parameters for this study. Figure 3 shows key measures from a single simulation run with the RR scheduling discipline. The

TABLE I
Path Terminology and Parameters

N	Set of possible mmWave paths	$\text{length}(N) = 8$
P	Paths available to the scheduler	$P \subset N$
B	Paths busy sending a packet	$B \subset P$
F	Paths free for use	$F \subset P$, $F \cup B = P$, and $F \cap B = \emptyset$
f_1	Free path that has been in F longest	$f_1 \in F$
$f_{\max}$	Free path with highest capacity	$f_{\max} = \max(f \in F)$
A	Application's required sending rate	2 Gbps
C	Current combined capacity of P	$C = \sum C^{(p)}$, $\forall p \in P$, $C^{(p)} \in \{C^{(p)}_{\text{LoS}}, C^{(p)}_{\text{NLoS}}\}$

TABLE II
Key Simulation Parameters Relevant to This Article. See [4] for a Detailed Description of the Simulator.

Channel path loss model	UMi [6]
Packet size	1500 B
Predictive Control Interval	150 ms
Queue CDF prediction time horizon	200 ms
Shadow fading time	20 ms
Average time in LoS	LinRange(3,4,length(N)) s
Average time in NLoS	LinRange(4,3,length(N) s
Simulation length	305 s (first 5 s discarded)
Number of independent repetitions	50

Salient points: (i) The random sequences for each channel and delays are all separate and independent so that they are identical regardless of the scenario. (ii) LoS↔NLoS transitions are modelled by a two-state continuous time Markov model with averages as in the table (as in [7]). (iii) Each run is initialised with a different set of seeds. (iv) We model the random shadow fading variations while keeping our receiver at a fixed distance from all base stations. This somewhat artificial scenario allows us to focus on the scheduling and ERC dynamics without having to account for receiver motion and changing base station distances.

[1] Our purpose is to understand the system dynamics rather than choose an optimal scheduler; something that will depend on architectural details of an implementation.

[2] See table I for the notation used in the paper regarding paths.

[3] We have also tried other values such as 1%, which yields similar results. The advantage is in not sending on the very slow, occasionally even 10^6 times slower, paths.

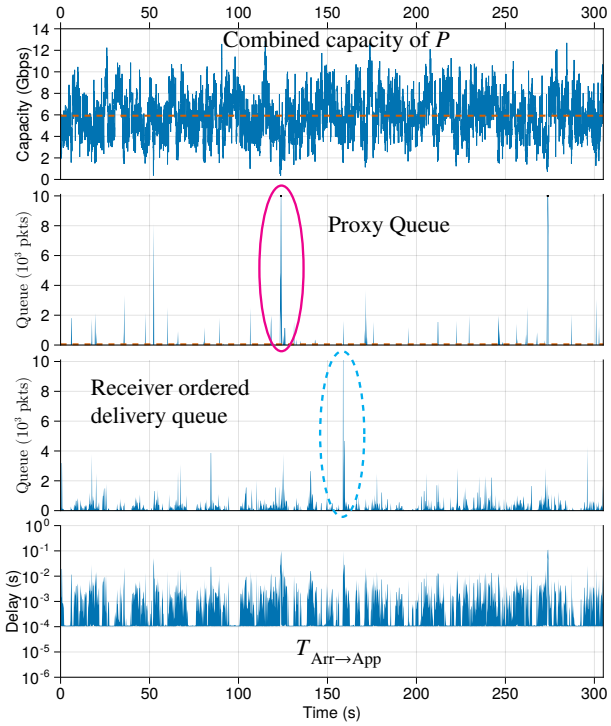


Fig. 3. RR scheduling example showing the combined capacity changes of the set of used paths, queueing at the two queues and the delay from proxy arrival to the application receives the in-order packet. The red dashed line indicates the average capacity and average proxy queue size.

predictive path manager [3, 4] attempts to ensure that there is enough capacity available to send the 2 Gbps of data by adding additional paths when needed and removing them when they are not needed. The resulting capacity from the continually varying number of used paths is shown in the Capacity graph at the top. Occasionally there is not enough available capacity, which results in proxy queue spikes that can be seen in the Proxy Queue graph (see magenta ellipse), the dots at the top representing packet losses in the proxy queue. The receiver queue graph shows the queueing at the receiver due to HoL blocking caused by out of order packets, such as when a packet is sent on a very low bit rate path while subsequent packets are sent on much faster paths (see dashed cyan ellipse). The final graph shows $T_{\text{Arr}\to\text{App}}$, the packet delay from when a packet arrives at the proxy until it is sent in-order to the receiver application. A good scheduler can reduce packet transit times from the proxy and significantly reduce HoL blocking delays at the receiver queue.

Figure 4 shows a comparison of the high percentiles of the CDF of $T_{\text{Arr}\to\text{App}}$ in the absence of any erasure correction. Both HCF and HCF+ noticeably improve over the simple RR discipline. The HCF+ discipline offers significant improvements to HCF in the 97–98 percentiles, pushing the delays down to under 200 μs. However, HCF+ is not work conserving, and the cost can be seen when the green dash-dot HCF line crosses the HCF+ red dashed line at around 1 ms.

III. Convolutional Packet Erasure Codes to Avoid Head of Line Blocking

The dynamic nature of the capacity of each path invariably leads to intermittent queues at the proxy, and infrequent severe

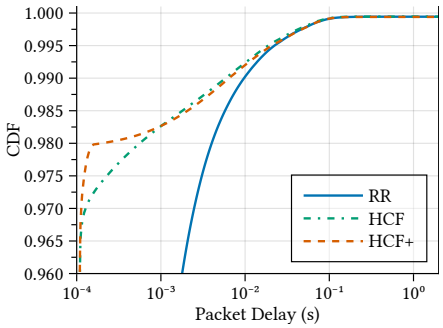


Fig. 4. Comparison of mmWave path selection using RR, HCF and HCF+ scheduling disciplines.

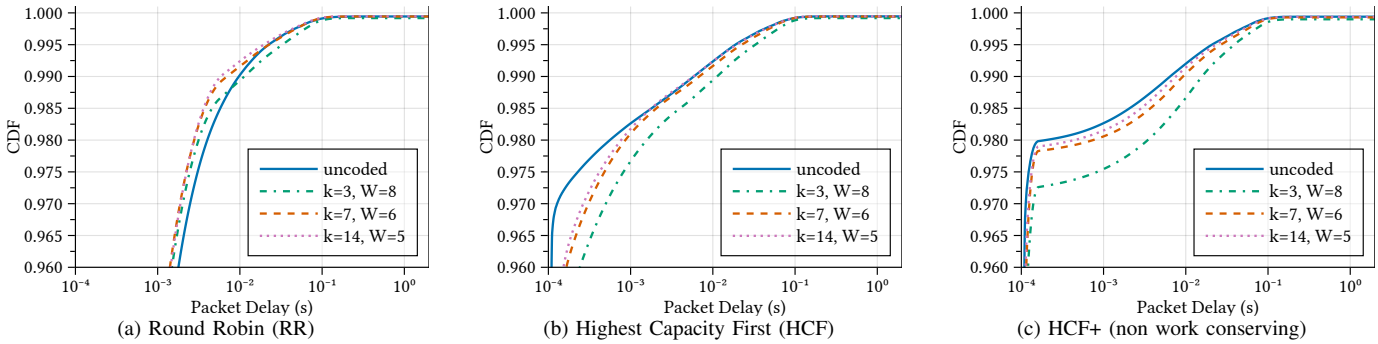
packet transit delays in some paths. For latency-sensitive applications, the effect of these path delays is similar to packet erasures. When this occurs, it makes sense to apply erasure-reconstruction codes (ERCs) to avoid receiver queueing delays. ERCs have properties similar to forward-error correcting codes (FECs) [5], but require simpler receiver processing since erasure locations are known. ERCs are used extensively in distributed cloud storage [8], Internet packet recovery [9–11], and coding for stragglers in distributed computing [12]. In each of these cases, codes are selected to optimize application specific criteria that fail to align perfectly with what we want for our application. Codes in [9] are designed for optimum efficiency in coding and decoding as well as a flexible adaptation to the Internet channel, but these codes work best with a very large block length, which implies long delays and problems with adapting to quick capacity changes. Convolutional codes [5] work by encoding few information symbols (here: packets) at a time, which allows a quick recovery at the receiver. Random linear convolutional codes are discussed in [10, 11]. In general, random codes are flexible, but offer sub-optimum performance and require extra overhead in capacity as well as extra processing at the proxy and the receiver.

Instead, for our application, we use ERCs designed to satisfy conditions 1) and 2) below. A set, P, of paths is made available to the scheduler by the path manager described in [4] (see Fig. 2 and table I). Packet traffic is scheduled onto these paths according to the scheduling disciplines described in section II. Coded communication entails adding a parity packet after every k information packets. This parity packet is a linear combination of the k information packets. The *sequence* of parity packets should satisfy the conditions:

1) After one block of k+1 packets has been sent, the receiver should be able to reconstruct all the k corresponding information packets as soon as *any* k packets have been received. More generally, after $w \geq 1$ blocks of $k+1$ packets each have been transmitted, the receiver should be able to reconstruct all the wk corresponding information packets as soon as *any* wk packets have been received.
2) The code should guarantee 1) for all $w \leq W$, where the window size W is as large as possible.

It turns out that for a *binary* convolutional code, the W value described in 2) is limited to 1. However, increasing the size of the finite field over which the convolutional code is constructed allows larger values of W. In this paper we consider the

This article has been accepted for publication in IEEE Communications Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2023.3269526



(a) Round Robin (RR) (b) Highest Capacity First (HCF) (c) HCF+ (non work conserving)

Fig. 5. Scheduling with convolutional erasure codes. The uncoded line corresponds to the case where no erasure codes are used, as in Fig. 4.

convolutional codes designed and presented in [13]. These codes are constructed over finite fields $\mathbb{F}_{2^m}$, $m \leq 14$, and have maximum or the highest value known for W for the field sizes considered.

There is a tradeoff in using coding: when paths are *correctly* estimated to provide sufficient capacity, coding is unnecessary and wastes capacity. In our application, coding can be seen as an "insurance" against unexpected channel capacity fluctuations, at a premium of extra capacity usage.

We now test the effect on the queuing delay from using these convolutional erasure codes with the three scheduling disciplines and the simulation model introduced in section II. The minor *computation* delay incurred by the recovery process of delayed packets is ignored, since this amounts to solving a typically very simple set of linear equations [13]. The path capacities in the simulator's Path Manager's predictive MMFQ based model are reduced by a factor $\frac{k}{k+1}$ to account for the extra parity packets. The erasure codes do indeed eliminate any significant queueing at the receiver's ordered delivery queue, but this does not necessarily translate to a corresponding delay advantage, as seen in Fig. 5. Erasure codes improve latency if reliable capacity information is not available or not used, such as in the RR scenario (see Fig. 5a): A bad scheduling choice (i.e., choosing a slow path for transmission) can be mitigated by use of an erasure code. However, when good path scheduling choices are made, prioritizing use of higher-capacity paths, the delay cost in the overhead outweighs the advantages of removing HoL blocking at the receiver. Even though the path manager attempts to preemptively provide enough capacity, this does not always happen: (i) the queue-state prediction may be incorrect; (ii) If all 8 possible paths are blocked (NLoS), the combined capacity may not be sufficient. When this happens, i.e., $\frac{k+1}{k}A > C$, queueing at the proxy causes more delay than is saved by removing HoL blocking.

This suggests sending parity packets only when $\frac{k+1}{k}A \leq C$. Of course, then some head-of-line blocking will occur, but since the delay impact of that is less than that of queueing at the proxy queue we expect to see an overall benefit. Figure 6 shows that this provides delay advantages for all three scheduling disciplines. Notice that the order of the erasure code lines on the CDF is the reverse of that in Fig. 5. This simple change to the schedulers compensates for the disadvantages of the higher overhead codes.

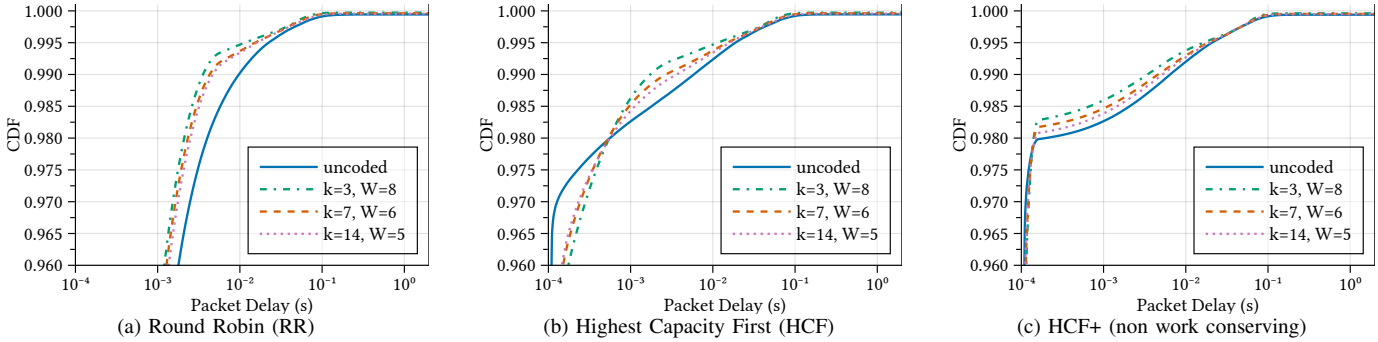
IV. Imperfect Path Capacity Knowledge

In the preceding results, the path manager and the schedulers had perfect knowledge of the capacity of the available paths. In a real system, this knowledge is likely to be delayed. Hence, next we introduce a delay in the proxy for obtaining capacity change information. We model this as a random truncated normally distributed delay, with parameters μ ms, $\sigma = \mu/10$, and limited by $[0, 2\mu]$. Fig. 7 presents results for interesting points of Fig. 6: $P\left[T_{\text{Arr}\to\text{App}} \leq \{300\,\mu\text{s}, 3\,\text{ms}\}\right]$, for $0 \leq \mu \leq 30$ ms, showing the trend as the working knowledge of the capacity becomes more stale. In a deployed system, the critical value will be determined by the overall sender to receiver delays and the jitter buffer configuration at the receiver, which is beyond the scope of this work.

The knowledge of path capacities affects the following three elements of the system: (i) the path manager's parametrization of the MMFQ and its selection of paths to add or remove; (ii) the path scheduling choices of HCF and HCF+; (iii) the choice of whether to send a parity packet or not. Imperfect capacity knowledge affects the RR discipline results the least since it does not use capacity knowledge to make scheduling decisions (Fig. 7a). Both HCF (Fig. 7b) and HCF+ (Fig. 7c) rely on knowledge of the channel capacity to schedule packets, with delays in capacity knowledge naturally affecting the smaller delay probabilities ($T_{\text{Arr}\to\text{App}} \leq 300\,\mu$s) more. Since the ERC scenarios suffer all three effects of the delayed capacity knowledge, they suffer more as indicated by the larger slopes of their respective lines compared with the uncoded scenario. The effect is most dramatic for HCF+, which relies on capacity knowledge for choosing not to send on very low capacity paths; the detrimental effects of the delay being exacerbated by it not using capacity when it is available. The effect of increasing delays in capacity knowledge on delay probabilities is roughly linear in these experiments suggesting that it will be easy to model and predict. This needs further verification, but was found to roughly hold for $P\left[T_{\text{Arr}\to\text{App}} \leq [0.3, 100]\,\text{ms}\right]$.

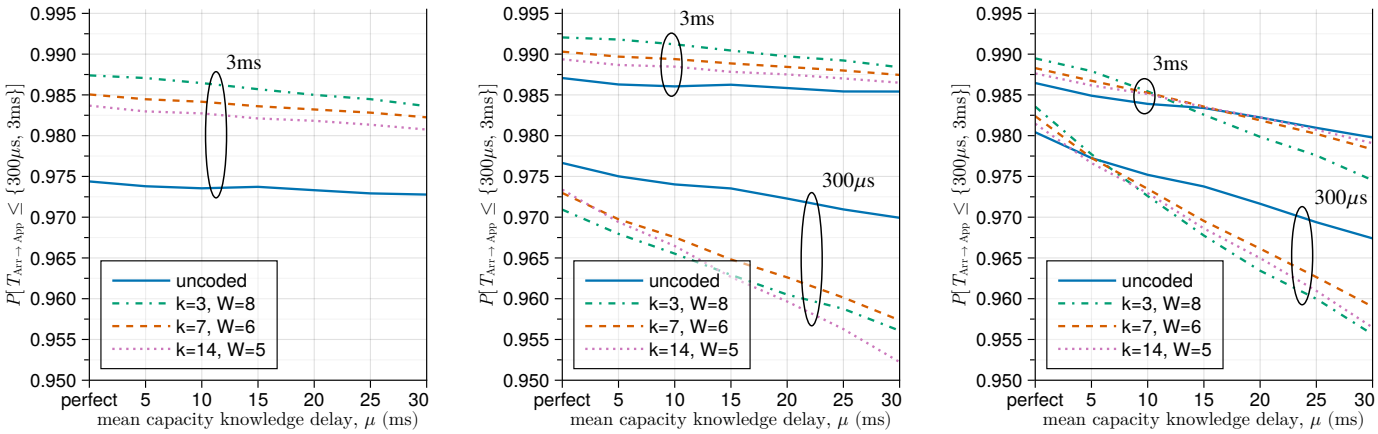
V. Conclusions and Future Work

The HCF and HCF+ capacity aware scheduling disciplines significantly reduce proxy to receiver delays, with the non-work conserving HCF+ allowing a high probability of very low delays. Adding ERC eliminates most HoL blocking delays in the receiver, but the overhead tends to increase the overall



(a) Round Robin (RR) (b) Highest Capacity First (HCF) (c) HCF+ (non work conserving)

Fig. 6. Scheduling with convolutional erasure codes, but sending parity packets only when capacity allows it, i.e., only when $\frac{k+1}{k} A \leq C$.



(a) Round Robin (RR). *Note* $T_{Arr\to App} = 300\,\mu s$ *is not in the range of probabilities depicted.* (b) Highest Capacity First (HCF) (c) HCF+ (non work conserving)

Fig. 7. Scheduling with imperfect capacity knowledge for $0 \leq \mu \leq 30$ ms. Sending parity packets only when capacity allows it.

delay, only having a net delay improvement in the naive RR scheduling scenario. Since the delay cost in sending parity packets is higher than the delays they reduce at the receiver queue, choosing not to send parity packets when capacity is insufficient redresses the balance in the capacity aware disciplines. This allows a net delay reduction at the expense of a little receiver queueing for HCF and HCF+ with ERC.

We investigated the effect of imperfect knowledge of path capacities in terms of delays in capacity knowledge. Looking at $P\left[T_{\text{Arr}\to\text{App}} \leq \{300\,\mu\text{s}, 3\text{ms}\}\right]$, we find the effect is roughly linear and affects the capacity aware schedulers more, and the non-work conserving HCF+ worst. This suggests that the benefits of HCF+ may not be achievable in a deployed system with a low ms target delay.

The effectiveness of ERC in reducing delay depends on the probability of packets being sent on very low capacity paths, which is related to the LoS↔NLoS transition probabilities. We plan to investigate this information theoretic problem to determine the ideal coding rates for any given scenario. Further steps in this work will examine the whole system [see 4], optimising in real-time the balance of mmWave path use, sender data rate, receiver quality, battery usage, scheduling and ERC coding for a particular scenario in use.

REFERENCES

[1] T. K. Vu, C.-F. Liu, M. Bennis, M. Debbah, and M. Latva-aho, "Path selection and rate allocation in self-backhauled mmWave networks," in *Proc. IEEE WCNC*, 2018, pp. 1–6.

[2] M. G. Dogan, M. Cardone, and C. Fragouli, *Proactive resilient transmission and scheduling mechanisms for mmWave networks*, 2022. arXiv: 2211.09307 [cs.IT].

[3] D. A. Hayes, D. Ros, Ö. Alay, and P. Teymoori, "Reliable consistent multipath mmWave communication," in *Proc. of ACM MSWiM*, Nov. 2021, pp. 149–158.

[4] D. A. Hayes, D. Ros, Ö. Alay, P. Teymoori, and T. M. Vister, "Investigating predictive model-based control to achieve reliable consistent multipath mmWave communication," *Computer Communications*, vol. 194, pp. 29–43, 2022.

[5] S. Lin and D. J. Costello Jr., *Error Control Coding*, 2nd. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.

[6] *5G; Study on channel model for frequencies from 0.5 to 100 GHz*, 138 901, v16.1.0, ETSI, Nov. 2020.

[7] G. R. MacCartney, T. S. Rappaport, and S. Rangan, "Rapid fading due to human blockage in pedestrian crowds at 5G millimeter-wave frequencies," in *Proc. IEEE GLOBECOM*, 2017, pp. 1–7.

[8] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[9] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[10] B. Adamson *et al.*, *Taxonomy of Coding Techniques for Efficient Network Communications*, RFC 8406, Jun. 2018.

[11] N. Kuhn, E. Lochin, F. Michel, and M. Welzl, *Forward Erasure Correction (FEC) Coding and Congestion Control in Transport*, RFC 9265, Jul. 2022.

[12] S. Li and S. Avestimehr, *Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning*. Now, 2020, vol. Foundations and Trends in Communications and Information Theory: Vol. 17, pp. 1–148.

[13] Á. Barbero and Ø. Ytrehus, "Rate $(n-1)/n$ systematic memory maximum distance separable convolutional codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3018–3030, 2018.