OASIS: Collaborative Neural-Enhanced Mobile Video Streaming

Shuowei Jin
University of Michigan
jinsw@umich.edu

Ruiyang Zhu
University of Michigan
ryanzhu@umich.edu

Ahmad Hassan
University of Southern California
ahmadhas@usc.edu

Xiao Zhu[*]
University of Michigan
shawnzhu@umich.edu

Xumiao Zhang
University of Michigan
xumiao@umich.edu

Z. Morley Mao
University of Michigan
zmao@umich.edu

Feng Qian
University of Southern California
fengqian@usc.edu

Zhi-Li Zhang
University of Minnesota
zhzhang@cs.umn.edu

Abstract

Neural-enhanced video streaming (*e.g.,* super-resolution) is an ongoing revolution which can provide extremely high-quality video streaming services breaking the restriction of bandwidth. However, such enhancements require intense computation power that is not affordable for a single mobile device, which hinders their real-world deployment. To address the limitation, we propose OASIS, the first system that facilitates multiple users in close proximity to execute intense neural-enhanced video streaming in real-time. To this end, OASIS intelligently distributes computation tasks among multiple mobile devices, selects appropriate video bitrates and super-resolution models, and optimizes video chunk delivery. As a result, the expensive neural-enhanced streaming is done through distributed collaboration, achieving optimal quality of experience (QoE). We implement and evaluate OASIS on commodity smartphones from different vendors, under various network and computation conditions. Extensive experiments demonstrate the high efficiency of OASIS: it improves the video streaming QoE by 40%-200% and reduces each participant's energy consumption by 60% when the system scales up from a single device to six devices.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → *Computer vision*; • **Human-centered computing** → **Ubiquitous and mobile computing**.

Keywords

Video streaming, mobile computing, super-resolution, deep neural networks

ACM Reference Format:
Shuowei Jin, Ruiyang Zhu, Ahmad Hassan, Xiao Zhu, Xumiao Zhang, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2024. OASIS: Collaborative Neural-Enhanced Mobile Video Streaming. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3625468.3647610

1 Introduction

Mobile video streaming has been prevalent for years, consuming a predominant majority of mobile data [38, 39, 43]. For instance, YouTube, one of the most popular video streaming services, reports that 70% of its views come from mobile devices, and the average mobile viewing session is over 40 minutes [3]. Users are craving for increasingly higher resolution video content. However, streaming high resolution video to mobile devices is challenging due to limited and fast varying wireless bandwidths [30, 37, 38]. Despite advances in adaptive bitrate (ABR) algorithms [12, 16, 17, 25, 36, 43], user-perceived quality-of-experience (QoE) still largely depends on the available network bandwidth [15, 26, 31, 32], which can be difficult to predict.

Neural-enhanced video streaming leverages client-side computation to overcome the bandwidth limitation problem [21, 41]. Specifically, super-resolution (SR) models are trained to learn a mapping from low-quality to high-quality video content. The client can fetch low-quality video content along with SR models from the server and apply SR inference to generate high-quality video content, reducing the impact of network bandwidth fluctuations while improving the user-perceived QoE. However, due to the computation-intensive nature of SR inference, current neural-enhanced video streaming systems are typically designed for powerful clients equipped with high-end GPUs, making them unsuitable for mobile devices with limited computation power [10, 41, 42]. Certain existing studies [40, 44] apply SR to only a portion of the video on mobile devices. However, these approaches remain constrained by the computational power of a single mobile device, thereby presenting opportunities for enhancement.

In response to these limitations, we explore *collaborative neural-enhanced video streaming for mobile devices*, a new design space that exploits both the computation power and network resources of multiple mobile devices in proximity to improve video streaming QoE. While previous studies [20, 45] have explored collaborative video streaming, they do not consider neural enhancement in their designs and only target network-level collaboration, *i.e.,* aggregating the network bandwidth of multiple devices to download video content. Along with network-level cooperation among devices, collaborative neural-enhanced video streaming also requires computation-level collaboration (leveraging multiple devices to run computationally intensive neural enhancement models). However,

[*] Now at Google.

MMSys '24, April 15–18, 2024, Bari, Italy
© 2024 Association for Computing Machinery.
ACM ISBN 979-8-4007-0412-3/24/04…$15.00
https://doi.org/10.1145/3625468.3647610

realizing a collaborative neural-enhanced mobile video streaming system faces new challenges. (i) It demands complex coordination of tasks, such as server-to-device video chunk downloading, device-to-device chunk forwarding, SR inference, and post-SR video distribution. (ii) Utilizing the full potential of devices with diverse network and computational resources is a complex yet essential undertaking. Maximizing the overall system performance while accounting for these differences remains an ongoing challenge. (iii) Achieving near-optimal ABR performance in a collaborative neural enhancement setting is also much harder than the single-device scenario or collaborative streaming without neural enhancement.

To tackle these intriguing challenges, we propose OASIS, a collaborative neural-enhancement system that enables users in proximity to stream high-quality video content on their devices cooperatively. OASIS is designed to handle two distinct use cases:

- Multi-user, multi-device scenario, is designed to enable a group of individuals to aggregate their network and computational resources for high-quality video streaming in locations with limited network connectivity. The content can be displayed on their personal devices, or on a larger screen with one phone connected to a TV.
- Single-user, multi-device scenario, caters to the growing trend of individuals owning multiple phones, including older models left unused at home. With our system, users can fully harness the capabilities of these devices to facilitate video streaming.

In OASIS, each device downloads a set of chunks (*i.e.,* part of the entire video content) from the server. Upon receiving these chunks, each device performs SR inference to upsample some of them to a higher quality, and forwards the remaining of them to other devices for SR, depending on their computation power and local wireless links' bandwidths. The upsampled chunks are then distributed to other devices so everyone has high-resolution video content.

We design two principled algorithms for OASIS: OASIS-ABR, which selects the optimal combination of resolution and SR model, and OASIS-SCHED, which schedules chunk downloads and SR inference tasks across devices. We decompose OASIS this way because we first need an ABR algorithm to decide the amount of total data the entire system handles (*i.e.,* the quality version hence data size of each video chunk to download), and then determine how much data each participating device handles (*i.e.,* scheduling). We focus on video-on-demand (VoD) in this work while OASIS can be extended to live video streaming.

OASIS-ABR enhances traditional ABR video streaming algorithms [23, 34, 43] by smartly selecting both video chunk resolutions and super-resolution (SR) models. It uniquely addresses multi-device collaboration, simplifying the system with a novel throughput prediction model. We also redesign the multi-armed contextual bandit framework, which enables the algorithm to deftly balance exploration (evaluating new ABR decisions) with exploitation (honing decisions based on existing knowledge), leading to more intelligent QoE decisions. OASIS-SCHED manages the downloading, local processing, or forwarding of these chunks based on each device's capabilities. This includes optimizing data flows and chunk assignments to minimize delays and enhance QoE.

Our experiments on up to seven mobile devices reveal that OASIS significantly improves video streaming QoE (by 35% to 230% over other systems), reduces energy consumption as more devices are added (up to 60% less), and surpasses traditional bitrate selection methods in QoE improvement (20% to 129%). Specifically, OASIS-SCHED greatly reduces stall times (by 75% to 100%) across devices with different computing and network resources.

2 Motivation and Use Cases

2.1 Incentives for Multi-device Collaboration

The rise of multi-device usage has led to innovative features for seamless collaboration. Apple's Continuity feature allows tasks to transition smoothly across Apple devices like Mac, iPhone, and iPad [8]. Similarly, Samsung, OPPO, and Huawei have developed features enabling users to manage and share resources across multiple devices, enhancing user experience [6, 7]. Prior research [20, 28, 45] has focused on designing network-level collaboration schemes to enhance throughput and reduce energy consumption.

OASIS draws inspiration from these mobile collaboration systems, but extends the concept further by leveraging deeper collaboration at both *network* and *computation* levels to facilitate high-quality video streaming across different versions of mobile devices.

We next list a few concrete use cases of our proposed system.

- **Bus Trip**: Friends on a bus trip watch a movie together on their phones, passing areas with poor cellular connectivity. They may subscribe to different carriers. It has been shown that under mobility, the performance of different carriers are not correlated [27]. The collaboration can help smooth the overall Internet capacity.
- **Group Tourism**: Foreign tourists wish to learn about the local culture and history through high-quality videos. They want to reduce cellular data usage because they use roaming data plans.
- **Rural School**: Students there with limited Internet access watch educational videos as part of their curriculum.
- **Solo Travel**: A traveler has a smartphone with cellular access and a more powerful tablet without Internet access. Our system allows the two devices to collaborate: the smartphone fetches the content, which is then upsampled by both devices.

2.2 Mobile SR Performance Measurement

For smooth video streaming, mobile devices must super-resolution (SR) *exceeds* the standard video frame rate of 24-30 FPS. Our analysis across five flagship smartphones: Samsung Galaxy S10 5G (S10), Google Pixel 5 (PX5), Samsung Galaxy S20 Ultra 5G (S20U), Samsung Galaxy S21 Ultra 5G (S21U), and Samsung Galaxy S22+ 5G (S22+), shows none meet such SR speeds threshold. We download {180p, 360p} video chunks from a university-hosted CDN server, and perform frame-by-frame SR to upscale the video to {720p, 1080p}. We leverage the smartphone GPU for SR processing and utilized the Diggers model [18], which is the fastest DNN model within our exploration, for video super-resolution. The upscaled video chunks are displayed using Google ExoPlayer. For performance evaluation, we record the SR processing time of each video chunk and present the results in Fig. 1. Furthermore, we measure the average performance of various SR models comparing with baseline methods (Resize, Bi-linear interpolation), as summarized in Table 1. An analysis of the figure and table reveals that higher input resolutions yield improved SR outcomes for the same target resolution, at the expense of increased inference latency. This is because a larger input data size provides more information to the SR model, leading to higher quality and a longer time for inference.

From Fig. 1, we can observe that, due to limited computational capabilities, no single device can deliver SR processing at a speed

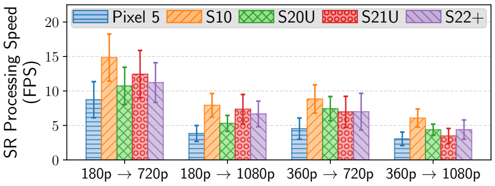


Figure 1: SR processing speed of latest phones.

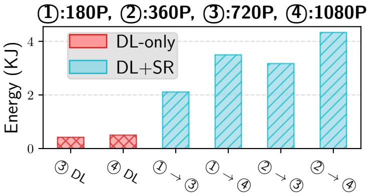


Figure 2: Energy overhead of SR for S10.

Table 1: Performance on PSNR and SSIM.

Resolution	Average PSNR/SSIM		
	Resize	*Bi-linear*	**SR**
180p-720p	27.19/0.79	28.27/0.83	**30.67/0.88**
180p-1080p	26.88/0.78	27.97/0.82	**30.32/0.87**
360p-720p	31.46/0.90	32.66/0.92	**35.68/0.95**
360p-1080p	30.58/0.87	32.07/0.90	**34.73/0.94**

exceeding 15 FPS on average. Even with significant advancements in mobile chipsets in the future, the complexity and computational demands of cutting-edge SR models are expected to rise concurrently. For example, the current leading Video SR model [24] can only achieve 2.778 FPS on an NVIDIA TITAN RTX GPU [9], a GPU that is 47 times more powerful than the one in a Pixel7 Pro [5]. This indicates that even with the advancement of mobile GPUs, the necessity for a collaborative system will remain to ensure a high-quality mobile video streaming experience.

The energy demands of deep neural network-based video super-resolution (SR) are significant, especially for mobile devices with limited battery capacity. To measure this, we hook an S10 phone to an external Monsoon power monitor and measure the energy overhead of the state-of-the-art mobile video SR model Diggers [18], enhancing video resolutions from 180p and 360p to 720p and 1080p. We also log energy when simply downloading the video chunks (*DL-only*). After subtracting the screen energy consumption, we plot the energy overhead for all six settings in Fig. 2. The results indicate that depending on the SR *input-output* combination, an S10 consumes 2.1-4.3kJ for a 5 minutes video. In other words, a 30-min SR video will result in 920-1920 mAh energy consumption on S10, which translates into a drastic 28-57% battery drain. Later results (§5.2.3) will demonstrate that a device's SR energy consumption can be lowered up to 60% by facilitating collaboration among smartphones. The collaboration will not only reduce the energy overhead for mobile devices but also improve the video streaming experience.

3 System Design

In this section, we first provide an overview of OASIS, our proposed collaborative mobile system for neural-enhanced video streaming (§3.1). Next, we describe OASIS's key design features: OASIS-ABR algorithm for adaptive bitrate and SR model selection (§3.2), and OASIS-SCHED algorithm for cross-device chunk scheduling (§3.3).

Given that displaying on a single device is a subset of the scenario where each device displays the content, our subsequent design will primarily focus on the system where each device needs to receive the post-SR chunk for individual display.

3.1 System Overview

OASIS enables collaborative neural-enhanced video streaming by effectively utilizing mobile devices' available network and compute resources. As shown in Fig. 3, OASIS systematically manages multiple modules to improve the video QoE. The **adaptive, resource-aware scheduler** is responsible for (i) adaptively selecting a video bitrate and SR model for all devices collectively and (ii) scheduling chunk download and SR tasks across the devices. Next, mobile devices execute the assigned tasks using the *Processor* module. Finally, the *Distributor* module allows a device to share downloaded and/or SR chunks with other devices. OASIS performs all the aforementioned tasks iteratively. The *Scheduler* acts as OASIS's brain and makes all the decisions in each iteration (*e.g.,* device A downloads chunk 1 at 180p, runs SR inference to upsample the chunk to 720p, and broadcasts it to all devices for streaming). Then all devices execute the assigned tasks, collect performance metrics (*e.g.,* network bandwidth and SR inference speed), and send the metrics back to the *Scheduler*. Using this information, the *Scheduler* devises an optimal execution plan for the next iteration.

The need for iterations in OASIS arises due to the constraints of traditional single-device adaptive video streaming systems, which determine the next chunk's bitrate based on current buffer size and predicted network bandwidth [23, 43]. This single-chunk approach is not well-suited for multi-device neural-enhanced video streaming, often leading to protracted chunk processing and idle periods for devices. To mitigate this, OASIS integrates both bitrate and SR model selections in each iteration, thereby allocating the processing of multiple chunks across different devices. The number of chunks to process in each iteration is determined based on the processing speeds of devices. The goal is to ensure the processing on each device finishes at roughly the same time. If the devices are homogeneous, we can just set the number of chunks in each iteration as the number of devices, and assign one chunk to each device. However, given the heterogeneity of devices, we determine the number of chunks as the least common multiple of each device's processing speed(unit: chunks/second). For the first iteration where we don't have device processing speed information yet, we set the number of chunks to process as $\lambda \times$ deviceNum, where λ is a predefined factor and is optimally set to 5 based on our experiments. Once all chunks in current iteration are downloaded, SR-enhanced, and distributed, the system advances to a new iteration.

Fig. 3 illustrates the architecture of OASIS. There are three types of components in OASIS: (i) a media server to store multi-bitrate video chunks and SR models, (ii) agent devices that run the *Processor* and *Distributor* modules, and (iii) a controller device that runs the *Scheduler* along with other OASIS modules. Although OASIS uses the device with the most powerful chipset as the controller by default, the choice of controller device is configurable [1]. Upon startup, the controller device establishes a WiFi hotspot (WLAN) for communication with agents. Each device connected to the hotspot uses Network Service Discovery (NSD) [4] to broadcast its presence and discover other devices. Through NSD, devices obtain the IP addresses and port numbers of their peers, enabling them to establish peer-to-peer direct socket connections for efficient data transfer and communication. The devices with Internet access also maintain a network connection (WWAN) with the media server. OASIS framework is flexible and allows devices with no Internet

[1] In our current implementation, users choose the controller device. The user initiating the collaborative streaming session can select his most powerful device as the controller. We leave as future work the automated controller election mechanism which evaluates the processing capacities of participating devices and select the most suitable one.

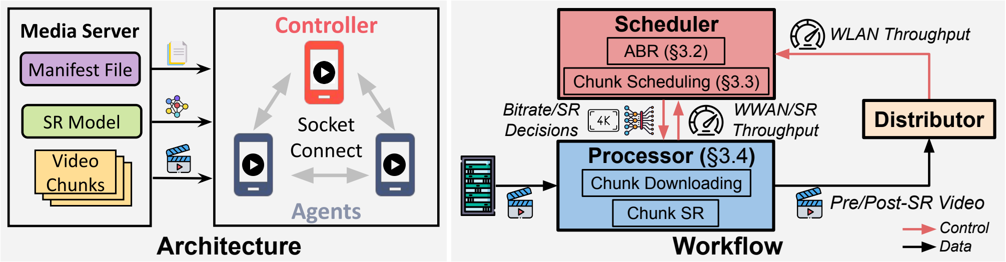


Figure 3: OASIS system architecture and workflow.

connectivity to collaboratively stream videos out-of-the-box. Next, we briefly discuss the overall workflow of OASIS.

Offline Video-content Preparation. Similar to DASH [1], after a video is uploaded, the media server first encodes the video at multiple bitrates (*i.e.,* tracks) and divides it into equally-sized chunks. Additionally, the media server trains the SR models using low-bitrate video tracks as inputs and high-bitrate tracks as ground truth. Finally, it generates a manifest file describing the available bitrate tracks and SR models for adaptive streaming. The file size of each model is less than 300KB.

Task Scheduling. Recall that the *Scheduler* mainly handles two major tasks: (i) It decides the bitrate to download from the media server and the SR model to employ for chunk SR based on the performance metrics (WWAN/WLAN bandwidth, computation speed) of all devices. The *Scheduler* achieves bitrate selection and SR model selection through a multi-armed contextual bandit algorithm OASIS-ABR (§3.2). It makes optimal bitrate and SR model decisions by intelligently exploring unchosen bitrate and SR model, making smart decisions. Due to its explainable and robust throughput modeling method, it can accurately predict system performance, thereby enabling optimal decision-making.

(ii) After OASIS-ABR comes up with a joint bitrate and SR model choice, each chunk is assigned to device/s for downloading and SR inference. For each downloaded chunk, the devices process it locally or forward it to the responsible agent for its SR inference. After the SR inference, each device broadcasts the upsampled chunks to other devices for streaming. The *Scheduler* uses our chunk scheduling algorithm OASIS-SCHED to distribute the download and SR inference tasks across all agent devices (§3.3). OASIS-SCHED minimizes stall time in video streaming by implementing a two-step scheduling process: first, it optimizes data flow across devices to enhance end-to-end throughput, and second, it allocates SR tasks to devices in a way that prioritizes earlier chunk completion times, consequently reducing system stall time. OASIS-SCHED reduces the stall time during video streaming by first scheduling the data flow across devices to improve the overall end-to-end throughput of the system, and then scheduling SR tasks to devices to ensure earlier have earlier finish time.

Task Processing. Once the *Scheduler* assigns download and SR inference tasks to a device, it utilizes the *Processor* module to execute these tasks. The *Processor* also collects performance metrics during task execution and sends them to the *Scheduler*. The *Scheduler* uses the collected metrics as inputs for OASIS-ABR and OASIS-SCHED algorithms. While downloading the video chunks from the media server, OASIS measures the WWAN network throughput. Our system also measures the WLAN network throughput when chunks are forwarded from one agent to another. Lastly, OASIS logs the SR inference speed of each device to estimate the instantaneous computation speed.

Chunk Distribution. The *Distributor* module enables sharing of video chunks across devices in two ways: (i) Once a device downloads a chunk from the server, the *Distributor* checks if the chunk's SR inference is assigned to other devices. If so, the *Distributor* forwards the chunk to the device responsible for its SR inference. Otherwise, the chunk's SR inference is conducted locally, and (ii) once the *Processor* upsamples a chunk through SR inference, the *Distributor* broadcasts the SR chunk to all devices for streaming.

3.2 Multi-armed Contextual Bandit Algorithm for Adaptive Bitrate Control

3.2.1 Problem Formulation Given the parameters defined in Table 2, our objective is to design an ABR algorithm f. This algorithm selects the optimal **combination** of download bitrate R_j and SR model M_j to maximize the QoE based on several key parameters. T_W and T_L represent the available bandwidths for streaming over WWAN networks and WLAN connections, respectively. S signifies the processing speed of each SR model, while B denotes the system's buffer status. Formally, the problem can be expressed as:

$$f(T_W, T_L, S, B) = (R_j^*, M_j^*) = \arg\max_{R_j, M_j} QoE(R_j, M_j) \qquad (1)$$

where QoE is calculated for a given bitrate R_j and SR model M_j. [2]

Traditional ABR algorithms [25, 43] are designed for single-device streaming and select bitrates accordingly. OASIS, however, deals with the added complexity of multi-device neural-enhanced streaming, presenting two main challenges:

- **Concurrent Selection of Bitrate and SR Model.** Choosing both the bitrate and SR model simultaneously adds complexity due to their interdependence. Also, the variability in quality enhancements offered by different SR models further complicates the decision.
- **Multi-device Collaboration.** Devices in a multi-device setup have varying capabilities, introducing additional complexity through the need for coordinated control and data sharing. The optimization process must account for this heterogeneity and the intricacies of collaboration to enhance the streaming experience.

In addressing these complexities, our first step is to better model the multi-device system, aiming to distill its intricacies and enhance

[2] The QoE, adopted from prior work [25, 40, 41, 43], combines video quality, smoothness, and rebuffering impact. Defined as $\frac{1}{N}\left(\sum q(R_n) - \mu \sum T_n - \frac{1}{N-1}\sum |\Delta q(R_n)|\right)$, it reflects the utility of bitrate R_n, rebuffering time T_n, and quality variation $\Delta q(R_n)$ across chunks. For adjustments due to super-resolution (SR), an inverse mapping from quality to bitrate is employed, aligning the bitrate to the quality enhancement similar to the approach in NAS and NEMO [40, 41].

Table 2: Common notations used in the algorithm.

Notation	Definition
$\hat{T}, T$	Predicted, actual system total throughput list for combinations.
T_W, T_L	WWAN and WLAN bandwidth list for devices.
M	SR Model list.
S	DNN inference speed nested list for models.
S_{M_j}	DNN inference speed list of models M_j across devices.
B, R	System buffer occupancy and download bitrate decision.
N_d	Number of participant devices in the system.
N_i	Number of chunks to be processed in each iteration.
α	Exploration parameter.
C, O	Counter list, overhead list for combinations.
L	Network latency list for devices.

its predictability of throughput performance[3]. With a more accurate understanding of the system's performance, we can then make informed decisions on the optimal bitrate and SR model. This two-pronged approach, refining the system model and then utilizing it for decision-making, serves as the cornerstone of our OASIS-ABR to optimize user experience.

3.2.2 System total throughput performance modeling.

This section delves into the modeling of the system's total throughput performance, utilizing the predicted values of T_W, T_L, and S_{M_j} when model M_j is chosen for SR. Note while devising effective throughput predictors for current T_W, T_L, and S_{M_j} based on historical data is a compelling research direction in its own right [46], our paper primarily focus on bitrate adaptation algorithms. We employ a heuristic approach, using the average of the previous five timestamps values to predict the current timestamp value.

Given various components of OASIS, a direct black-box model prediction of system total throughput based on T_W, T_L, and S_{M_j} might be deficient in terms of explainability and robustness. Real-world deployments often diverge from training datasets. Variations in the number of devices or their characteristics can significantly influence system performance. Consequently, we dissect the system's total throughput into two segments: a theoretical throughput upper bound and the overhead from multi-device interactions.

- **Upper bound: bottleneck determines performance.** Our system workflow consists of raw chunk downloading, SR processing, and post-SR chunk distribution in a pipelined manner. Overlooking multi-device collaboration overhead, the system can be perceived as a singular entity with aggregated WWAN, SR processing speed, and WLAN metrics. In such a pipeline, the system's efficiency is inherently limited by its slowest segment. This insight allows us to use $\min\{\sum_{i=0}^{N_d} T_{W_i}, \sum_{i=0}^{N_d} T_{L_i}, \sum_{i=0}^{N_d} S_{M_{j_i}}\}$ as theoretical upper bound.
- **Overhead modeling.** In practical scenarios, due to scheduling algorithm imperfections, the system seldom achieves its theoretical upper bound, and overhead invariably persists. We define this overhead as the gap between the theoretical upper bound and the actual throughput. An overhead list, O, is constructed to track historical overheads for all bitrate and SR model combinations: (R_j, M_j). Each combination's overhead (O_j) is computed by averaging its overhead values from previous iterations.

With both the overhead and throughput upper bound values at hand, throughput can be estimated as shown in Equation 2. This method mitigates errors stemming from discrepancies between training and real-world datasets, ensuring swift real-world adaptability. Additionally, it enhances system transparency and explainability, with overhead values providing performance insights. To ensure robustness, we can overestimate the overhead to fit potential network or computational fluctuations.

Algorithm 1: OASIS-ABR algorithm.

```
Input  : $O, S, T_W, T_L, \alpha$
Output : combination choice, $(R_j, M_j)$
for each combination choice $i$ do
    /* Predict end-to-end system throughput.                          */
    $\hat{T}_i \leftarrow$ PredictThroughput$(O_i, S_{M_i}, T_W, T_L)$;
    /* Estimate QoE for $i^{th}$ combination.                          */
    $Q\hat{o}E_i \leftarrow$ EstimateQoE$(\hat{T}_i, B)$;
    /* Calculate UCB (upper confidence bound) value.                  */
    $UCB_i \leftarrow QoE_i + \alpha\sqrt{\frac{\log(\sum C_i+1)}{C_i+1}}$;
end
/* Select combination with highest UCB score.                         */
Select combination ID: $j = \arg\max_i UCB_i$;
/* Output $j^{th}$ combination $(R_j, M_j)$, wait for runtime measured
   system total throughput, SR inference speed to update.             */
$T_j, S'_{M_j} \leftarrow$ Runtime measurement results;
/* Update overhead, SR inference speed of $j^{th}$ combination by
   averaging newly measured and historical values.                    */
$O_j \leftarrow O_j + \frac{1}{C_j}((T_j - \hat{T}_j) - O_j), S_{M_j} \leftarrow S_{M_j} + \frac{1}{C_j}(S'_{M_j} - S_{M_j})$
/* Update combination counter and total counter.                      */
$C_j \leftarrow C_j + 1, \mathrm{C} \leftarrow \mathrm{C} + 1$;
```

$$\hat{T}_i = \min\{\sum_{i=0}^{N_d} T_{W_i}, \sum_{i=0}^{N_d} T_{L_i}, \sum_{i=0}^{N_d} S_{M_{j_i}}\} - O_i \tag{2}$$

3.2.3 Algorithm details.

With our throughput prediction module in place, a straightforward approach for the ABR algorithm would be to evaluate every combination of download bitrate and SR model.

However, a challenge arises when trying to determine the SR models' inference speed on different phones. One method is profiling each model's speed at the start of streaming, but as indicated in §2.2, this is time-consuming. A more practical approach is to measure the model's inference speed during the streaming process. This approach, however, presents a dilemma: without testing various SR models, the complete SR model inference speed information is hard to collect, making it difficult to identify the optimal choice. Sticking to a single ABR decision (exploitation) could lead to suboptimal QoE. On the other hand, continuously exploring new models might strain the system's computational resources. Thus, finding the right balance between exploration and exploitation remains a challenge.

To adeptly address the exploration-exploitation dilemma, we introduce OASIS-ABR, as detailed in Algorithm 1. The algorithm begins by initializing the QoE, counter, and overhead for each combination choice, followed by setting the exploration parameter, α. The counter tracks the number of times each bitrate and SR model combination is selected and will be used to calculate the exploration factor (second term in Equation 3).

In each iteration, the algorithm predicts throughput, estimates QoE for each combination, and calculates an upper confidence bound (UCB) value for each option. The UCB, defined in Equation 3, balances the known rewards' reliability (exploitation) with the potential of discovering more advantageous options (exploration). The UCB value combines the expected QoE (first component) with a exploration factor (second component). The exploration term of a combination decreases as the frequency of its selection (C_i) increases, thus encouraging exploration of less chosen combinations. The combination with the highest UCB is then selected for

[3] We define system throughput as the total amount of data processed by all devices in the system over a specified period of time, divided by the processing time.

Algorithm 2: OASIS-SCHED algorithm.

```
Input  : $T_W, T_L, S, N_d$;
Output : Sched_WWAN, Sched_WLAN, Sched_SR;
/* Start Data Flow Scheduling.                                      */
/* Model the system as a graph. Generate adjacency matrix.          */
$G_{input}[0,i] = T_{W_i}, G_{input}[i,1] = ... = G_{input}[i,N] = T_{L_i}, G_{input}[N,i] = S_i$;
/* Filter to directed graph's adjacency matrix.                     */
$G_{directed}[i,j] = \begin{cases} 0, & if(i == j)or(G_{input}[0,i] < G_{input}[0,j]), \\ G_{input}[i,j], & if(G_{input}[0,i] >= G_{input}[0,j]) \end{cases}$;
/* Apply Ford-Fulkerson algorithm to find the maximum flow.         */
$G_{flow}, N_{capacity} = FordFulkerson(G_{directed})$;
/* Start Chunk Scheduling                                           */
/* Initialize devices' WWAN Request/WLAN Request/SR Lists.          */
Sched_WWAN$_i$, Sched_WLAN$_i$, Sched_SR$_i$ $\leftarrow \{\}, \{\}, \{\}$;
/* Predict network latency of each device based on $G_{flow}$.      */
$L \leftarrow$ PredictNetworkLatency($G_{flow}$) ;
/* Create predicted completion time queue for each device           */
$Complete_i = \{L_i + 1/S_i, L_i + 2/S_i, .., L_i + N_i/S_i\}$
for $i \in [1, ..., N_d]$ do
    $j = \arg\min_i Complete_i[0]$;/* Find minimum completion time  */
    Sched_SR$_j$.$append(i)$; /* Assign $Chunk_i$ to $Device_j$     */
    $Complete_j.dequeue()$;
end
Sched_WWAN, Sched_WLAN$\leftarrow$ PostProcess($Sched\_SR, G_{flow}$);
```

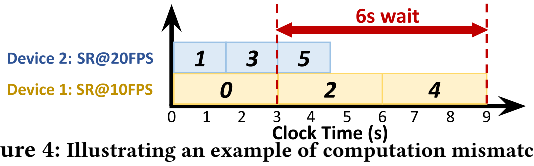


Figure 4: Illustrating an example of computation mismatch.

execution.

$$UCB_i = \hat{QoE}_i + \alpha\sqrt{\frac{\log(\sum C_i + 1)}{C_i + 1}} \tag{3}$$

After execution, the algorithm updates based on the actual total throughput and SR inference speed collected in runtime. Unlike traditional multi-armed bandit algorithms that rely on direct rewards (e.g., QoE), our approach updates the system's internal throughput predictor model by averaging the updated overhead (O_j) for the jth combination and the SR inference speed (S_{M_j}) for the chosen model (M_j). The system dynamically updates each device's WWAN and WLAN bandwidths in every iteration, enabling precise throughput predictions. This adaptability ensures accurate QoE estimations in the UCB term, optimizing decision-making under fluctuating network conditions. The evaluation utilizing real-world variable network traces in § 5.2 demonstrates the robustness of our approach.

3.3 Network and Computation Co-Scheduling

Once OASIS-ABR determines the best bitrate and SR model combination per iteration, the controller device allocates chunk downloading tasks among all devices. It then sends the scheduling results to the agents, detailing which video chunks to download, super-resoluted, or forward to other devices for SR. Effective scheduling is vital; naive methods can degrade system performance. For instance, in Fig. 4, with two devices of differing processing speeds (20 FPS and 10 FPS), a round-robin scheduling causes the faster device to wait 6 seconds after processing chunk 3, highlighting the need to consider device computation heterogeneity during scheduling.

The OASIS-SCHED is designed to optimize system performance in two distinct but complementary ways. At a high level, it focuses on optimizing the data flow across devices to address the heterogeneity in network and computation resources and to maximize the system's end-to-end throughput, thus reducing the average stall time to improve QoE. On a detailed level, the algorithm performs chunk scheduling by prioritizing the completion time of earlier chunks to reduce stall time, taking into account each path's network delay and computation delay. By optimizing both the data flow and chunk completion times, the algorithm ensures that the system performs at its best and provides a high-quality user experience. We demonstrate the workflow of OASIS-SCHED in Fig. 5 and the pseudo-code in Algorithm 2.

3.3.1 Data Flow Scheduling In a heterogeneous system, achieving maximum end-to-end throughput necessitates a scheduling approach that accounts for the diverse WLAN and WWAN bandwidths and computational capabilities of each device. Our solution transforms this challenge into a graph optimization problem, offering a structured way to represent devices, network bandwidth, and computational resources.

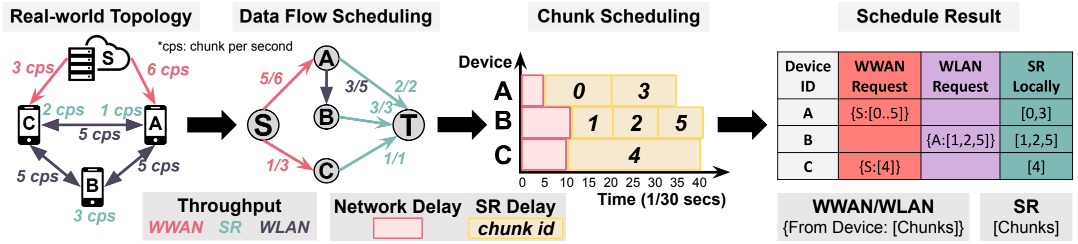
We model the system as a graph demonstrated in Fig. 5 from real-world topology to data flow scheduling part, where each device is a node. Two additional nodes are introduced: a source node representing the server and a target node symbolizing the post-SR broadcasting procedure. The edges in this graph have specific real-world interpretations and capacities:

- The edge between the source node and a device node represents using WWAN to download from the server. Its capacity is defined by the device's WWAN bandwidth, indicating the maximum number of chunks that can be downloaded in unit time.
- Edges between device nodes signify the chunk forwarding procedure, where a device uses its WWAN bandwidth to assist another device in downloading and then forwards the chunk for processing. The capacity of these edges is determined by the WLAN bandwidth.
- The edge from a device node to the target node represents the SR procedure. The SR speed, which dictates the maximum number of chunks ready for broadcasting, determines its capacity.

Note, in our modeling, there is no edge to represent the post-SR broadcasting procedure. Instead, the target node alone symbolizes this phase. This design choice is rooted in practical considerations: typically WLAN bandwidths set up by the hotspot far exceed the bandwidth requirements for broadcasting. As a result, the throughput limitations in the post-SR broadcasting phase become negligible.

To further optimize, we transform our flow graph into a directed graph. We retain only edges from devices with higher WWAN bandwidth to those with lower bandwidth, based on the fact that devices with higher throughput are more capable of assisting the lower-throughput devices. In this graph structure, maximizing system throughput translates to maximizing the flow from the source node to the target node. Once the directed graph is established, we utilize the Ford-Fulkerson method [13] to find the maximum flow from the source node to the target node. This method searches for augmenting paths – paths where additional flow can be sent – and increases the flow along these paths until no more augmenting paths exist. In the context of our system, an augmenting path represents a sequence of data transfers from the server to the devices for SR and ultimately broadcast to all devices for display. The algorithm maximizes the use of available network and computational resources and enhancing the video streaming experience..

3.3.2 Chunk Scheduling After data flow scheduling, the next challenge is chunk scheduling, which assigns specific chunk IDs to



Device ID	WWAN Request	WLAN Request	SR Locally
A	{S:[0..5]}		[0,3]
B		{A:[1,2,5]}	[1,2,5]
C	{S:[4]}		[4]

Figure 5: OASIS-SCHED Workflow.

each device for SR. The essence of video streaming is the sequential display of chunks. Hence, ensuring earlier chunks are processed before later ones is important. A simple example illustrates this: if a video player awaits the display of the 5^{th} chunk, but the system processes the 6^{th} chunk first, a stall ensues.

To address this, our algorithm predicts the completion time of each chunk on every device, accounting for both network and computation delays. Having the predicted completion time queue, we assign the earlier chunk to the earlier finishing time slot judiciously as demonstrated in the chunk scheduling part of Fig. 5.

After determining the SR tasks for each device and combining this information with the data flow graph, we generate a comprehensive WWAN/WLAN request list for all devices. For example, as demonstrated in the schedule result part of Fig. 5, if device B need to do SR on chunks 1, 2, and 5, the data flow scheduling graph suggests that device A should forward these chunks to device B via WLAN. This results in generating both WLAN and WWAN requests for the respective devices.

As the system operates, each device continuously measures its WWAN throughput, WLAN throughput, and SR processing speed. At the end of each iteration[4], devices send these metrics to the controller, which then updates the graph structure and parameter list for the next iteration, ensuring a dynamic and responsive system. This iteration process continues until all video chunks have been processed and distributed, marking the completion of the streaming.

4 Implementation

We implement OASIS on commodity mobile devices supporting the HLS streaming protocol with 12K lines of code (LOC). Conceptually, OASIS's design is not protocol-dependent and can be applied to any adaptive streaming protocol. For offline SR model training, we modify the source code of Diggers [18], which is trained using the MegEngine framework. The trained SR model is converted to a TensorFlow Lite Model using TensorFlow 2.4.1 for inference on commodity Android devices.

We implement the mobile-side collaboration system using ExoPlayer for video display, and OpenCV and FFmpeg for decoding video chunks into images and encoding super-resolved images back into video chunks. To achieve higher inference speed, we execute the SR model in GPU delegate mode. OASIS's design and implementation offer flexibility, enabling it to support inference for other neural-enhanced video editing models (e.g., video frame interpolation), thereby increasing OASIS's usability.

To optimize end-to-end throughput efficiently, we pipeline chunk downloading, SR processing, and chunk distribution tasks. We also pipeline decoding, SR, and encoding phases within the processing stage. Once a device receives a chunk, it decodes the video chunk into consecutive images, splits them into batches, and feeds them into the SR deep neural network (DNN). The SR DNN processes the images through its feature extraction block (FEB) module and generates high-resolution output images, which are then re-encoded into video chunks for transmission towards the target(s).

We also offer users a configurable choice to balance initial streaming quality and startup delay. Users can opt for either a longer startup delay (10-20s) for immediate high-quality SR video streaming or a shorter startup time (<1s) with initial non-SR video chunks for the first 10-20 seconds and after sufficient buffering, the system seamlessly transitions to SR video, accommodating the latency challenges outlined in §5.2.4.

5 Evaluation

5.1 Experimental Setup and Methodology

Video Dataset and SR Models. We selected 4K videos from the most popular categories on YouTube: *How-to*, *Vlogs*, and *Games*, picking the top ten most viewed videos supporting 4K at 30fps, limited to the first 5 minutes. These videos were transcoded into multiple bitrate versions (300, 800, 1800, 2500, 4300 kbps) across resolutions (180p, 360p, 540p, 720p, 1080p) with a chunk duration of 1 second. For SR model training, each video track was converted into an image dataset. We trained [5] four SR models for each video category, {180p → 720p, 180p → 1080p, 360p → 720p, 360p → 1080p}. Our SR model architecture follows Diggers [18], utilizing bidirectional RNNs with efficient feature extractors to capture temporal dependencies. The evaluation results suggest that our SR models have similar performance in all video categories. We selected a video from the *Games* category [2] for our experimental streaming.
System Parameters. For SR model training, we set the training epochs to 40 and the learning rate to 2×10^{-3}. We empirically set the exploration parameter α to 1.
Devices. We use seven smartphone devices in total: two Pixel5 (PX5) phones, one Samsung S10 (S10), three Samsung S20 Ultra (S20U) phones, and one Samsung S21 Ultra (S21U). We employ two Monsoon Power Monitors for energy experiments, which are attached to S10 and S20U to measure accurate variations in power consumption. During the energy consumption measurement, screen brightness is set to maximum. We host the videos on a university-hosted CDN server that supports both HTTP and HTTPS protocols.

[4] Throughout this paper, the iteration concept remains consistent. It refers to the cycle during which the system selects bitrates and SR models for the upcoming set of chunks, assigns these chunks for processing, and then all devices execute their assigned tasks.

[5] The SR models were trained using frames from higher resolutions as ground truths, for example, the 180p → 720p model was trained with 180p inputs and 720p outputs as the ground truth.

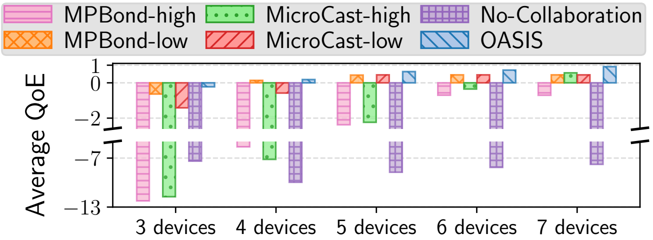
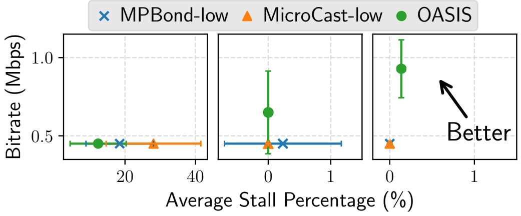


Figure 6: Average QoE across various device configurations.



(a) 3 devices (b) 5 devices (c) 7 devices

Figure 7: QoE breakdown.

Evaluation Metrics. We use the QoE metric as in [40, 41] and set the parameter $\mu = 4.3$ as the highest bitrate track (4.3Mbps) of the video dataset. This metric has been widely used in previous work [25, 40, 41, 43]. To further analyze the system performance, we break down the QoE into average bitrate and total stall time. During the energy experiment, the total system energy consumption is calculated by subtracting the device's screen energy consumption from the device's total energy consumption.

Baselines. We employ MicroCast and MPBond as baselines. MicroCast aggregates multiple devices' network resources for enhanced video download rates, while MPBond manages multipath transport for improved streaming efficiency. We integrate our SR processing pipeline into these baselines for fairness. Additionally, a no-collaboration scheme is used as another baseline where each device handles SR independently. Furthermore, in §5.2.2, OASIS's performance is evaluated against these baselines in download-only scenarios (excluding SR processing).

5.2 End-to-end Evaluation

We first evaluate the end-to-end QoE results of OASIS as the system scales up. Then, we evaluate how the energy consumption changes in OASIS devices as the system scales up. Throughout this section's experiment, we use real network traces used in Pensieve [25].

5.2.1 End-to-end QoE Analysis. To evaluate the performance of our proposed system, we compare it to two well-established mobile collaboration systems, MicroCast and MPBond. As these systems do not have their own ABR algorithms, we use two modes for comparison purposes, namely *low* and *high*. In *low* mode, the download bitrate is fixed at 180p and the SR model applied is 180p → 720p. In *high* mode, the download bitrate is set to 360p and the SR model used is 360p → 1080p. For the case where no collaboration scheme is in place, the ABR is set to *low* as the system lacks sufficient computational resources. In OASIS, the ABR algorithm selects from the aforementioned four SR models to optimize performance.

To thoroughly examine the system's performance as the number of devices increases, we constructed five unique scenarios, ranging from three to seven devices. As shown in Fig. 6, OASIS consistently achieves the best performance across all scenarios, enhancing the average QoE by 35% to 230% in comparison to the baselines. In every scenario, the no-collaboration approach yields subpar results since participants do not reap any benefits when the system expands. When fewer than 6 devices are present within the system, both MPBond-high and MicroCast-high suffer from inferior QoE performance due to insufficient computational resources. In a 7-device scenario, MicroCast-high manages to attain a positive QoE, albeit still 38% lower than OASIS.

In 3-5 device scenarios, both MPBond-low and MicroCast-low outperform MicroCast-high, MPBond-high, and No Collaboration baselines since the lower-quality models are more advantageous when computational resources are limited. Nevertheless, the QoE of MPBond-low and MicroCast-low does not improve when the system includes more than 6 devices. This stagnation can be attributed to the absence of an efficient ABR algorithm in the baseline systems. When computational resources surpass the requirements of low-quality SR, fixing the download bitrate and SR model offers no additional advantages. However, thanks to our OASIS-ABR, the QoE of OASIS continues to rise as the system scales up. In a 7-device setup, OASIS's QoE is 378% higher than in a 4-device scenario.

We delve further into the two primary components of QoE, bitrate and average stall percentage, in Fig. 7. We illustrate only MPBond-low and MicroCast-low, the two best-performing baselines, along with OASIS. In the 3-device configuration, due to the lack of computational resources, OASIS-ABR consistently chooses the low-quality bitrate (180p) and the SR model (180p → 720p). Despite making the same ABR decision as MPBond-low and MicroCast-low, OASIS achieves 37% and 63% less stall time than MPBond-low and MicroCast-low, respectively, due to its scheduling algorithm OASIS-SCHED, which results in a higher QoE.

We also observe that as the system scales up, OASIS-ABR prioritizes to SR higher bitrates, leading to an increased QoE for OASIS while incurring zero stalls. MicroCast-low and MPBond-low can also benefit from the system's expansion. However, due to an inefficient ABR algorithm, MicroCast-low and MPBond-low are unable to adapt to a higher-quality SR model as computational power increases. This limitation restricts their ability to fully capitalize on the potential improvements in QoE as the system scales.

5.2.2 Download-Only Experiments. To push the boundaries of OASIS, we compare it with MicroCast and MPBond in a download-only scenario (no SR), for which they were originally designed. We equip MicroCast and MPBond with the RobustMPC [43] ABR algorithm, allowing them to select from video tracks with resolutions of 180p, 360p, 540p, 720p, 1080p for downloading. In such a download-only context, we set the SR speed in the algorithm to the positive infinity values to remove the SR processing constraint from the system. In this case, OASIS will focus on optimizing the distribution of tasks and data flows solely based on network bandwidth, without considering the SR processing time.The QoE results, in Fig. 9, are derived through min-max normalization, applied to each chunk's QoE within the experiment. Across all configurations, OASIS surpasses the baselines by 23.2% to 38.3%, attributable to the more intelligent decisions made by OASIS-ABR and OASIS-SCHED. The results demonstrate OASIS's robustness and superiority in all scenarios.

5.2.3 Energy Consumption. We next conduct experiments to determine if the collaboration scheme in OASIS can effectively

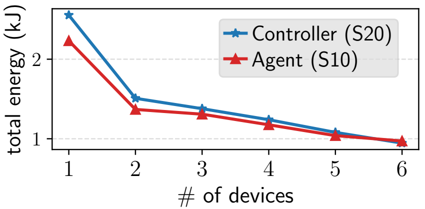


Figure 8: Energy consumption of participant devices when system scales up.

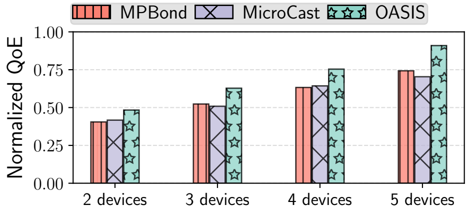


Figure 9: Average QoE in download-only mode at various device configurations.

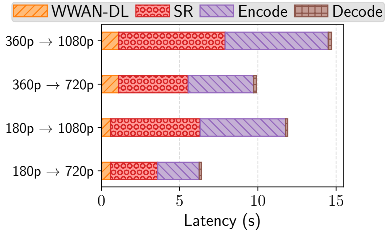


Figure 10: Latency breakdown.

reduce per-device energy consumption. To ensure a fair comparison, we modify the ABR selection result to consistently choose 180p video content and the (180p → 720p) SR model. We initiate the experiment with a single-device neural-enhanced video streaming scenario and gradually scale up the system. Throughout the multi-device experiment, we utilize an S10 linked to a power monitor as an agent device and an S20 connected to another power monitor as the controller to measure energy consumption. Fig. 8 displays the total system energy consumption of S10 and S20U for each of the six experimental settings. We observe that the energy consumption of both the controller and agent decreases as the system scales up, demonstrating that OASIS efficiently utilizes the computing power of all devices. Scaling up from 1 to 6 devices reduces the total energy consumption for S20 and S10 by 66.9% and 56.4%, respectively.

5.2.4 Latency breakdown Fig. 10 presents the breakdown of latency in processing a chunk for the entire streaming pipeline. We measure the latency for each component of the system: ABR, chunk scheduling, WLAN Download, Decode, SR, Encode, and WLAN Transfer. As ABR only takes 0.2ms, chunk scheduling requires a mere 0.3ms, and WLAN transfer averages 20ms, these values are relatively small compared to other components, and therefore, they are not displayed on the plot. The minimal values achieved by ABR and chunk scheduling indicate the efficiency of our algorithm and suggest that the overhead of our scheduling algorithm is quite low. The relatively low WLAN transfer time can be attributed to the high bandwidth provided by the controller's hotspot (*e.g.,* the hotspot established by S20U offers approximately 860 Mbps bandwidth).

Besides, we observe that the primary sources of latency are the SR and the encoding of post-SR images back into chunks. Comparing with the end-to-end processing latency derived from Fig. 1, the pipelined design successfully reduces the average latency by approximately 60%. This reduction demonstrates the effectiveness of our approach in optimizing the system's overall performance.

5.3 Ablation Study

We analyze the performance of each system component in OASIS and compare it with the state-of-the-art algorithms. As it is hard to control mobile devices' inference speed to measure the performance of each component under various computing power settings, we simulate the SR procedure by setting up a delay to control the inference speed throughout the ablation study experiments.

5.3.1 OASIS-ABR Evaluation. We perform experiments to understand the performance of OASIS-ABR under different computing power settings. We compare OASIS-ABR with state-of-the-art ABR algorithms: rate-based [23], buffer-based [17], RobustMPC [43]. As these algorithms only determine the download bitrate, we divide each algorithm into two baselines: *high* and *low*. *High* indicates that when the ABR algorithm chooses the download bitrate, it always chooses the highest SR model to use, while *low* indicates the contrary. For example, when RobustMPC algorithm decides to choose 180p video content to download, RobustMPC-High will upsample it to 1080p, while RobustMPC-low will upsample it to 720p.

During the experiment, we employ two devices in the system and set two computing power scenarios for evaluation.

- **25FPS**. In this setting, we set the highest quality SR model (360p → 1080p) inference speed on device 0 as 15 FPS, and device 1 as 10FPS. Using the inference speed ratio of different SR models obtained from earlier results (Fig. 1), we set each model's inference speed on each device accordingly.
- **30FPS**. We set the inference speed of the highest quality SR model on device 0 as 20 FPS and on device 1 to 10 FPS. Using the ratio of each model's inference speed from Fig. 1, we also set each model's inference speed accordingly.

We present the average QoE performance result in the left of Figure 11. In the 25FPS setting, the system lacks computation power to perform the highest quality SR. Consequently, bitrate selection plays a crucial part in the system. OASIS-ABR surpasses the baseline algorithms from 17% to 129%. From the middle of Figure 11, Rate-High and RobustMPC-High have the longest stall time as both baselines choose the highest quality bitrate and SR model without taking computing power into account. In Fig. 12, the QoE cumulative distribution function is presented. OASIS-ABR exceeds all the baselines. In the 30FPS situation, the left of Figure 11 demonstrates that OASIS-ABR still outperforms all the baseline algorithms by 5% to 69%. The system has enough computation power to perform the highest quality SR in this scenario. Thus, baselines such as RobustMPC-High and Rate-High which always choose the highest quality model achieve comparable performance to OASIS-ABR, however, OASIS-ABR reaches a shorter stall time.

5.3.2 OASIS-SCHED Evaluation. We evaluate the performance of OASIS-SCHED in the presence of heterogeneity in either computation or network bandwidth among devices. The performance of OASIS-SCHED is compared to the scheduling algorithms of MicroCast and MPBond using the video stall time metric. For the purpose of the ablation study, the ABR algorithm is fixed to select a 180p video content for download and SR to 720p.

Three devices are used, with a total computation speed of 33 FPS and a total network bandwidth of 6Mbps. In the computation heterogeneity setting, the network bandwidth is evenly divided among the devices, while the computation speed is divided in the ratios of 1:1:1, 2:1:1, 3:1:1, 4:1:1, and 5:1:1, gradually increasing the heterogeneity of computation speed across the devices. In the network heterogeneity setting, the computation speed is evenly divided among the devices, while the network bandwidth is divided in the

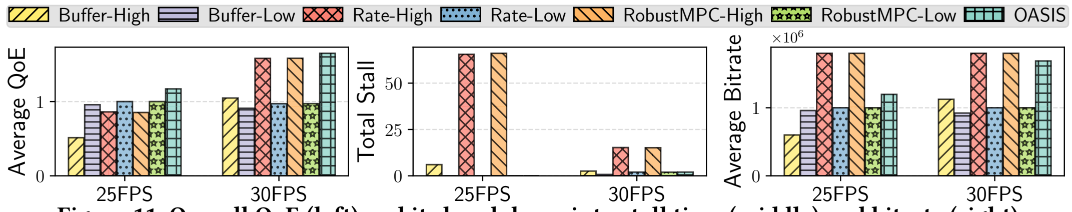


Figure 11: Overall QoE (left) and its breakdown into stall time (middle) and bitrate (right).

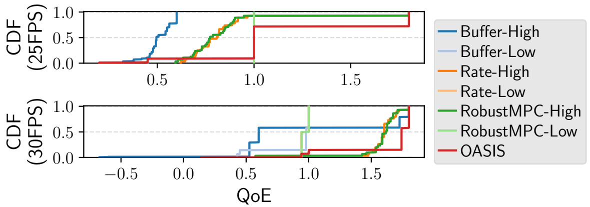


Figure 12: QoE CDF in 25FPS and 30 FPS setting.

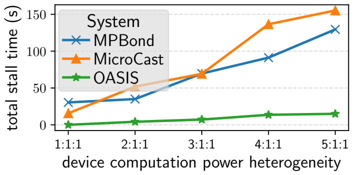


Figure 13: Performance change as computation heterogeneity increases.

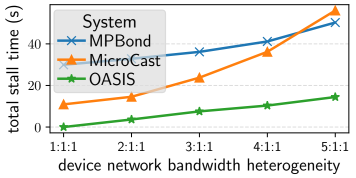


Figure 14: Performance change as network heterogeneity increases.

ratios of 1:1:1, 2:1:1, 3:1:1, 4:1:1, and 5:1:1, progressively increasing the heterogeneity of network bandwidth across the devices.

In the computation heterogeneity setting, as depicted in Fig. 13, OASIS outperforms the baseline algorithms by achieving the shortest stall time in all five scenarios. The results show that OASIS results in a reduction of 90% to 100% in stall time compared to the baseline algorithms. As the heterogeneity of computation increases, the total stall time for both MicroCast and MPBond increases significantly, while OASIS experiences only a slight increase. These results demonstrate that OASIS is capable of handling computation heterogeneity more effectively than the baseline algorithms.

In the network heterogeneity setting (Fig. 14), we can see that OASIS outperforms the baseline algorithms by achieving the shortest stall time in all five scenarios. The results show that OASIS results in a reduction of 75% to 100% in stall time compared to the baseline algorithms. As the heterogeneity of network increases, the total stall time for both MicroCast and MPBond increases significantly, while OASIS experiences only a slight increase. These results demonstrate that OASIS is capable of handling network heterogeneity more effectively than the baseline algorithms.

6 Related Work

Adaptive streaming. Adaptive streaming encodes video content at varied bitrates and divides it into equal-length chunks. Some ABR algorithms [17, 19, 34, 35] determine chunk bitrates based on network bandwidth and playback buffer. Advanced methods like MPC [43] and Pensieve [25] leverage control theory and reinforcement learning to optimize QoE, outperforming heuristic-based algorithms. However, they mainly cater to single-device scenarios.

SR-based video streaming system. NAS [41] is an SR-based video streaming system. It enables desktop SR streaming by replacing the low-resolution buffer with high-resolution post-SR chunks during video-on-demand (VoD) playback. Live-NAS [21] further extends NAS to support SR live streaming by performing SR after the uploading phase of live broadcasts. NeuroScaler [42] reduces the overhead of neural-enhanced live streaming by sharing SR DNNs across similar videos. Hyunho *et al.* [40] developed NEMO, an SR-integrated codec that accelerates SR execution on single devices, reaching 30 FPS, which is orthogonal to our collaboration theme.

Multi-device collaboration. MicroCast [20] enhances video streaming by pooling smartphone network resources, while MPBond [45] extends collaboration for content retrieval across personal mobile devices. Both primarily address network-level collaboration, leaving computational collaboration unexplored, which is crucial for our context. Kibbutz [28] shares links among nearby users to save energy, but it's limited to dual-device deployment. Cool-Tether [33] proposes WLAN utilization for web access on multiple devices. However, neither directly addresses computational resource sharing, which is central to our system. CloneCloud [11] introduces an architecture for elastic execution between mobile devices and the cloud, offloading computation to enhance performance and energy efficiency without manual partitioning. Furthermore, Femto Clouds [14] leverages mobile devices to provide cloud services at the edge, reinforcing the feasibility and effectiveness of utilizing ambient computation for mobile applications. PRISM [22] improves TCP performance using WWAN and WLAN, but its kernel-space solution isn't mobile-friendly. Mobile Plus [29] facilitates resource and functionality sharing across devices, but neither directly tackles computation sharing, a central theme of our work.

7 Conclusion

OASIS pioneers the realization of both network-level and computation-level collaboration, offering a unique perspective on device collaboration. It introduces a sophisticated end-to-end framework that efficiently harnesses the collective network and computational power of multiple devices. This collective power is leveraged to run neural-enhanced video streaming, a computation-heavy task. In real-world scenarios, OASIS's bitrate selection algorithm (OASIS-ABR) and scheduling algorithm (OASIS-SCHED) significantly outperform the baselines, improving video QoE by up to 129% and completely eliminating video stalls. Our work explores a new direction in multi-device collaboration, setting a precedent for future research.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported by the NSF under the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant 2112562, in addition to NSF Grants CMMI-2038215, CNS-1930041, CNS-2321532, CNS-2323174, CNS-2128489, CNS-2321531, NSF Award 1915122, and NSF Award 2128489.

References

[1] 2012. MPEG-DASH. https://dashif.org/.

[2] 2019. Red Dead Redemption 2: Official Gameplay Video. https://www.youtube.com/watch?v=Dw_oH5oiUSE/.

[3] 2020. 57 Fascinating and Incredible YouTube Statistics. https://www.brandwatch.com/blog/youtube-stats/.

[4] 2022. NSD. https://developer.android.com/develop/connectivity/wifi.

[5] 2023. *AI Benchmark Performance Ranking.* https://ai-benchmark.com/ranking.html

[6] 2023. *Expanding the Galaxy Ecosystem: Ultimate Connected Experiences Between Galaxy Devices.* https://news.samsung.com/us/samsung-expanding-galaxy-ecosystem-ultimate-connected-experience-between-devices

[7] 2023. *Huawei introduces a new era of cross-device collaboration with Super Device.* https://consumer.huawei.com/sg/press/news/2022/news-220223/

[8] 2023. *Use Continuity to work across Apple devices.* https://support.apple.com/guide/mac-help/work-across-devices-using-continuity-mchl1d734309/mac

[9] 2023. *Video Super-Resolution on MSU Video Super Resolution Benchmark: Detail Restoration.* https://paperswithcode.com/sota/video-super-resolution-on-msu-vsr-benchmark

[10] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. 2021. BasicVSR: The search for essential components in video super-resolution and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 4947–4956.

[11] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems.* 301–314.

[12] Mallesham Dasari, Kumara Kahatapitiya, Samir R Das, Aruna Balasubramanian, and Dimitris Samaras. 2022. Swift: Adaptive video streaming with layered neural codecs. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22).* 103–118.

[13] Lester Randolph Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* 8 (1956), 399–404.

[14] Karim Habak, Mostafa Ammar, Khaled A Harras, and Ellen Zegura. 2015. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *2015 IEEE 8th international conference on cloud computing.* IEEE, 9–16.

[15] Ahmad Hassan, Arvind Narayanan, Anlan Zhang, Wei Ye, Ruiyang Zhu, Shuowei Jin, Jason Carpenter, Z Morley Mao, Feng Qian, and Zhi-Li Zhang. 2022. Vivisecting mobility management in 5G cellular networks. In *Proceedings of the ACM SIGCOMM 2022 Conference.* 86–100.

[16] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. 2018. Favor: Fine-grained video rate adaptation. In *Proceedings of the 9th ACM Multimedia Systems Conference.* 64–75.

[17] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM.* 187–198.

[18] Andrey Ignatov, Andres Romero, Heewon Kim, and Radu Timofte. 2021. Real-time video super-resolution on smartphones with deep learning, mobile ai 2021 challenge: Report. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2535–2544.

[19] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies.* 97–108.

[20] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. 2012. Microcast: Cooperative video streaming on smartphones. In *Proceedings of the 10th international conference on Mobile systems, applications, and services.* 57–70.

[21] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication.* 107–125.

[22] Kyu-Han Kim and Kang G Shin. 2005. Improving TCP performance over wireless networks with collaborative multi-homed mobile hosts. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services.* 107–120.

[23] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733.

[24] Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. 2022. Vrt: A video restoration transformer. *arXiv preprint arXiv:2201.12288* (2022).

[25] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication.* 197–210.

[26] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. 2021. A variegated look at 5G in the wild: performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference.* 610–625.

[27] Yunzhe Ni, Feng Qian, Taide Liu, Yihua Cheng, Zhiyao Ma, Jing Wang, Zhongfeng Wang, Gang Huang, Xuanzhe Liu, and Chenren Xu. 2023. {POLYCORN}: Data-driven Cross-layer Multipath Networking for High-speed Railway through Composable Schedulerlets. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23).* 1325–1340.

[28] Cătălin Nicutar, Dragoş Niculescu, and Costin Raiciu. 2014. Using cooperation for low power low latency cellular connectivity. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies.* 337–348.

[29] Sangeun Oh, Hyuck Yoo, Dae R Jeong, Duc Hoang Bui, and Insik Shin. 2017. Mobile plus: Multi-device mobile platform for cross-device functionality sharing. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* 332–344.

[30] Devdeep Ray, Jack Kosaian, KV Rashmi, and Srinivasan Seshan. 2019. Vantage: optimizing video upload for time-shifted viewing of social live streams. In *Proceedings of the ACM Special Interest Group on Data Communication.* 380–393.

[31] Susanna Schwarzmann, Clarissa Cassales Marquezan, Marcin Bosk, Huiran Liu, Riccardo Trivisonno, and Thomas Zinner. 2019. Estimating video streaming QoE in the 5G architecture using machine learning. In *Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks.* 7–12.

[32] Susanna Schwarzmann, Clarissa Cassales Marquezan, Riccardo Trivisonno, Shinichi Nakajima, Vincent Barriac, and Thomas Zinner. 2022. Ml-based qoe estimation in 5g networks using different regression techniques. *IEEE Transactions on Network and Service Management* 19, 3 (2022), 3516–3532.

[33] Ashish Sharma, Vishnu Navda, Ramachandran Ramjee, Venkata N Padmanabhan, and Elizabeth M Belding. 2009. Cool-tether: energy efficient on-the-fly wifi hotspots using mobile phones. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies.* 109–120.

[34] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.

[35] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference.* 272–285.

[36] Xuedou Xiao, Wei Wang, Taobin Chen, Yang Cao, Tao Jiang, and Qian Zhang. 2019. Sensor-augmented neural adaptive bitrate video streaming on UAVs. *IEEE Transactions on Multimedia* 22, 6 (2019), 1567–1576.

[37] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. 2015. piStream: Physical layer informed adaptive video streaming over LTE. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* 413–425.

[38] Shichang Xu, Subhabrata Sen, Z Morley Mao, and Yunhan Jia. 2017. Dissecting VOD services for cellular: performance, root causes and best practices. In *Proceedings of the 2017 Internet Measurement Conference.* 220–234.

[39] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Alexander Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming.. In *NSDI*, Vol. 20. 495–511.

[40] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. NEMO: enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking.* 1–14.

[41] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18).* 645–661.

[42] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM 2022 Conference.* 795–811.

[43] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication.* 325–338.

[44] Zhengdong Zhang and Vivienne Sze. 2017. FAST: A framework to accelerate super-resolution processing on compressed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* 19–28.

[45] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2020. MPBond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services.* 364–376.

[46] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K Sinha. 2015. Can accurate predictions improve video streaming in cellular networks?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications.* 57–62.