DeepSFU: Scalable Deepfake Detection for Video Conferencing

Tuan Tran*, Shirin Ebadi*, S. M. H. Hosseini*, Woongsub Shin*, Evan Ram*,
Youngwook Son*,§, Seyeon Kim‡, Nam Bui¶, Kyunghan Lee§, Eric Keller*, Sangtae Ha*
*University of Colorado Boulder §Seoul National University ‡Korea University ¶University of Colorado Denver
tuan.tran, shirin.ebadi, hosseini, woongsub.shin, evan.ram, youngwook.son, eric.keller, sangtae.ha@colorado.edu,
seyeon625@korea.ac.kr, nam.bui@ucdenver.edu, kyunghanlee@snu.ac.kr

Abstract

Deepfakes have emerged as a significant threat to online communications, enabling nearly indistinguishable impersonation of executives, public figures, and trusted contacts during video calls. While state-of-the-art deepfake detection models can achieve high accuracy offline, deploying them in real-time video conferencing systems remains challenging: the added computation quickly violates interactive latency budgets and greatly limits scalability. Our empirical analysis reveals that video decoding and frame movement dominate the detection pipeline, together accounting for approximately 86.6% of per-frame processing time.

In this paper, we present *DeepSFU*, a novel video conferencing architecture that enables scalable deepfake detection as a native capability. *DeepSFU* reduces per-frame overhead by detecting anomalies directly on the encoded video bitstream, avoiding costly high-resolution video decoding and heavy detection model inference. To further improve scalability, *DeepSFU* offloads frequent packet encryption and data movement to Smart Network Interface Card (SmartNIC) by leveraging hardware-accelerated cryptography and GPUDirect RDMA. Intensive evaluation on a large-scale testbed shows that *DeepSFU* reduces per-frame processing latency by 143.7×, translating into 26.2× higher conferencing capacity while providing continuous deepfake detection for all participants.

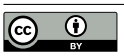
CCS Concepts

• **Networks** → **Programmable networks**; *In-network processing*; *Middle boxes / network appliances.*

Keywords

Video Conferencing, Deepfake Detection, Video Analytics, Selective Forwarding Unit, SmartNIC, WebRTC.

ACM Reference Format:
Tuan Tran, Shirin Ebadi, S. M. H. Hosseini, Woongsub Shin, Evan Ram, Youngwook Son, Seyeon Kim, Nam Bui, Kyunghan Lee, Eric Keller, and Sangtae Ha. 2026. *DeepSFU*: Scalable Deepfake Detection for Video Conferencing. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3789240.3829184



SIGCOMM '26, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/26/08
https://doi.org/10.1145/3789240.3829184

1 Introduction

Remote communication has shifted from voice-only calling to video-first interactions. In the early 2010s, adoption was limited: Pew Research Center reported that only 19% of U.S. adults had ever tried video calling or chat [47]. More recently, USC Marshall's Neely Social Media Index (2023) found that 30.4% of U.S. adults used FaceTime at least once in the 28 days preceding the survey [13]. Today, video calls are a core channel for work, education, collaboration, and healthcare [31, 33, 34, 43, 58, 62] at massive scale [48, 66].

In prior years, many scams relied on audio-only social engineering, or voice phishing, a serious worldwide threat. Attackers often impersonate authority figures (e.g., prosecutors, police, or bank officials) to create urgency and extract funds. In South Korea, reported voice-phishing losses were ₩447.2 billion in 2023 and had already exceeded ₩1 trillion (≈ US$675 million) by October 2025 [52, 53]. Europe has seen large-scale impersonation: Eurojust reports a Ukraine-based call-center network posing as police or bank employees, causing over €10 million in losses across 400+ known victims [12]. As video calling becomes routine, these tactics are evolving into deepfake scams that use AI-generated video to impersonate trusted individuals and pressure victims to transfer money or disclose sensitive information [15].

Deepfake scams are now widely reported. In one well-known case, attackers used a deepfake-enabled video meeting to impersonate senior executives and authorize transfers totaling HK$200 million (≈ US$25 million) [6, 16]. Deepfakes have also been used maliciously in video conferencing beyond corporate fraud [3, 11, 42, 49]. These incidents make real-time detection urgent and motivate native deepfake-detection-as-a-service within the conferencing stack, analogous to built-in cloud AI assistants (e.g., Zoom AI Companion), while addressing a broader societal need.

Current major video conferencing platforms (e.g., Zoom [67], Google Meet [17], and Microsoft Teams [32]) use a Selective Forwarding Unit (SFU) architecture, where centralized servers relay media among participants [1]. Unlike legacy Multipoint Control Units (MCUs) that decode, mix, and re-encode a composite stream [61], an SFU avoids transcoding and performs packet-level forwarding. It receives one stream per participant and selectively replicates and forwards packets to others based on network and device constraints [63]. This design scales to large meetings under tight latency budgets, but complicates server-side deepfake detection because the SFU does not operate on decoded frames.

Prior deepfake detection is either per-frame or temporal. Per-frame methods use heavyweight DNNs with high accuracy but high latency, while temporal methods are lighter but usually less accurate [2, 50, 51, 64]. Both approaches need high-resolution decoded frames, so naive SFU integration is costly: decode all ingress frames,

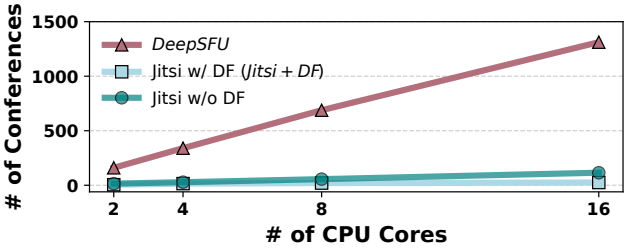


Figure 1: Maximum concurrent deepfake-detected conferences. Despite GPU decoder and extra CPU cores, Vanilla Jitsi (w/o DF) and *Jitsi+DF* scale poorly, while *DeepSFU* sustains continuous detection and supports 26.2× more conferences.

move them to GPU, and run inference at scale. These overheads easily violate latency budgets and cut SFU throughput, exposing a tension between packet-level forwarding and frame-level analysis, and motivating new designs that scale accurate detection without turning the SFU into a full transcoding pipeline.

To this end, we propose *DeepSFU*, an innovative SFU architecture specifically designed to enable large-scale, real-time deepfake detection. To avoid the major decoding bottleneck in detection pipelines, *DeepSFU* tracks facial manipulations directly in the encoded video bitstream. It cleverly leverages naturally embedded encoder information to identify changes across consecutive frames. This eliminates most of the compulsory high-resolution decoding load required by deepfake model inference. To further reduce data-movement overhead, *DeepSFU* transfers frame data directly from the SmartNIC to GPU memory, completely bypassing the host CPU. Finally, *DeepSFU* offloads per-receiver packet encryption to the SmartNIC's hardware crypto engine, enabling parallel execution and freeing CPU cycles for deepfake-related tasks.

Figure 1 shows the number of concurrent conferences supported by the de-facto open-source SFU Jitsi Videobridge (JVB [21]), as well as its optimized integration of a deepfake detection (DF) pipeline (*Jitsi+DF*), under a typical conference size of 15 users and a standard quality of 720p@30fps. Both Vanilla Jitsi and the DF-enabled variant (*Jitsi+DF*) fail to scale even with additional CPU resources and a GPU decoder, whereas our proposed solution *DeepSFU* scales exceptionally well, supporting 26.2× more conferences. By making deepfake inference efficient at the SFU, *DeepSFU* enables deepfake detection as a service within the conferencing infrastructure. The contributions of this paper are four-fold:

- We establish the first performance baseline for deepfake detection in video conferencing systems, *Jitsi+DF*. This enables quantification of media packet processing, frame decoding, and data-movement overheads.
- We propose a lightweight face-manipulation detector that analyzes encoded frames directly, triggering selective high-resolution decoding and model inference only when needed to reduce expensive computation on the common-case path.
- We design *DeepSFU*, a novel video conferencing system that enables scalable deepfake detection as a native SFU capability by orchestrating media forwarding and deepfake detection pipelines.
- We implement and evaluate *DeepSFU* on a large-scale experimental testbed, showing substantial throughput, latency, and scalability gains compared to *Jitsi+DF*.

2 Preliminaries

2.1 Selective Forwarding Unit

An SFU is a packet-level media bridge in video conferencing with a data plane operating on media packets and a control plane determining which quality each receiver should get.

Data plane. The SFU receives either *(i)* a single Scalable Video Coding (SVC) [59] stream embedding multiple quality layers (resolution/frame rate) under one stream identifier (SSRC), or *(ii)* multiple simulcast [4] streams at different qualities. The data plane parses packet metadata, exposes it to the control plane for quality selection, then selectively replicates and forwards packets matching each receiver's target quality.

Control plane. The bitrate controller estimates bandwidth per layer/stream from packet metadata and combines them with receiver feedback. Based on these signals, the control plane selects the layer/stream to deliver to each receiver, aiming to maximize perceived video quality while maintaining low latency and jitter.

2.2 Cryptography in SFU

Hop-by-hop (HBH) encryption. To secure communication, HBH encryption is the mandatory standard, where each media stream uses a unique key negotiated via DTLS. Each ingress stream is re-encrypted into distinct ciphertexts for different recipients. The SFU decrypts media to access plaintext, enabling SFU-side AI features (e.g., Zoom AI Companion).

End-to-end (E2E) encryption. Media payloads are encrypted with participant-only keys. HBH encryption still applies underneath, creating double-encrypted payloads that hide content from the SFU while preserving metadata [57] for forwarding. Thus, enabling SFU-side AI features requires disabling E2E encryption [68].

2.3 Baseline *Jitsi+DF* System

To establish a baseline for real-time deepfake detection in video conferencing, we extend JVB with an E2E pipeline, which we refer to as *Jitsi+DF*, where decoding, pre-processing, and model inference run on the GPU to minimize data movement (Figure 2).

Media-forwarding path. (1) *Jitsi+DF* receives and decrypts ingress media packets from the sender, each carrying a fragment of a VP9-encoded video frame. Upon decrypting media packets, (2) *Jitsi+DF* continues to replicate, re-encrypt and transmit egress media packets to receivers immediately, so deepfake detection runs off the critical forwarding path, minimizing media quality degradation. (3) For deepfake detection, *Jitsi+DF* aggregates these packets into complete frames while keeping the forwarding path non-blocking.

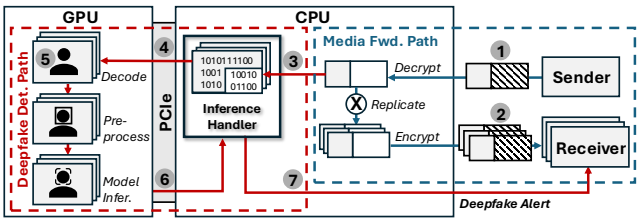
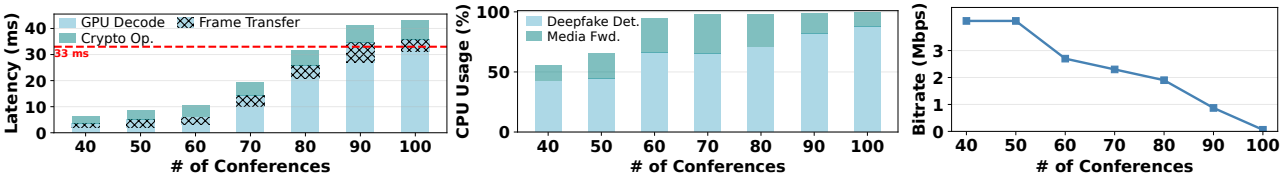


Figure 2: *Jitsi+DF* system overview. The media forwarding path processes and transmits media traffic, while the deepfake detection path reassembles video frames and runs end-to-end deepfake detection.



(a) Per-frame latency breakdown (b) CPU usage breakdown (c) Per-user media bitrate

Figure 3: *Jitsi+DF* profiling result. As concurrent conferences grow, CPU/GPU decoder/PCIe contention inflates decode, frame-transfer—together 86.6%—and cryptographic latencies pushing beyond the inter-frame budget (red dashed line). CPU contention with media forwarding further reduces delivered bitrate, degrading QoE.

Deepfake-detection path. Once a complete encoded frame is assembled, (4) it is transferred from host to GPU memory and (5) decoded using the GPU's hardware decoder (NVDEC [35]). The decoded frame is then pre-processed (e.g., NV12 to RGB color conversion, frame resizing) and fed into the detection model. (6) The classification result is informed back to the Inference Handler; if a deepfake frame is detected, (7) the system notifies all receivers.

3 SFU Deepfake Detection Insights

3.1 Real-Time Deepfake Detection Bottlenecks

To understand scalability limits of real-time deepfake detection, we profile *Jitsi+DF* under full provisioning (16 CPU cores, entire GPU) with typical conference settings: 15 users and standard media quality of 720p@30fps. Figure 3 reports the profiling results.

Non-scalable real-time decoding ((5)). Figure 3a breaks down per-frame processing latency and shows that real-time decoding quickly becomes the dominant bottleneck as the number of conferences grows: all ingress streams contend for the same GPU decoder resources. As decoding demand scales up, per-frame latency rises sharply and soon exceeds the 33 ms inter-frame budget.

Fragile decoding dependency ((5)). Exceeding this budget triggers a failure cascade. When frames arrive faster than decoding, they build up in a decode queue capped at 100 frames; beyond that, frames are tail-dropped. Because encoded video has temporal dependencies, dropping one inter-frame can corrupt the decoder state and block subsequent output. In our measurements (not shown), drops appear at 90 concurrent conferences (0.99%) and rise to 2.2% at 100 conferences; because drops affect any stream, 66 of 100 conferences fail to decode. Consequently, *Jitsi+DF* cannot produce valid frames for continuous deepfake detection.

Significant frame-transfer overhead ((4)). Increased concurrency amplifies PCIe contention, serializing transfers and increasing frame-transfer time (Figure 3a). Each transfer incurs host synchronization overhead, consuming CPU cycles rather than processing media packets (Figure 3b). This synchronization competes with media forwarding ((1) & (2)) for CPU resources, reducing forwarding capacity and causing bitrate degradation (Figure 3c).

Large cryptographic workload ((1) & (2)). Cryptography is the major overhead in the forwarding path due to per-receiver HBH encryption (§2.2). For N users, *Jitsi+DF* decrypts N ingress packets—one per uplink stream. However, it must replicate and re-encrypt media for every receiver with different keys, creating $N(N-1)$ egress copies. Although crypto latency is lower than frame decoding (Figure 3a), aggregate crypto work requires substantial CPU cycles; when co-located with deepfake detection pipeline, this significantly reduces throughput (Figure 3c).

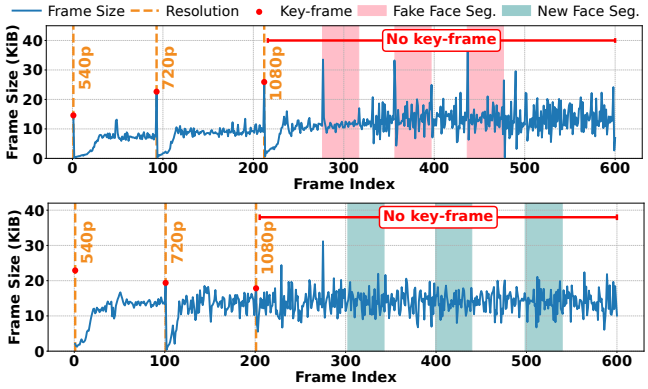


Figure 4: Frame dynamics: neither face-swaps of existing users nor newly appearing deepfake faces trigger additional key-frames.

3.2 Why Naive Sampling Fails

To study the impact of live deepfakes in video conferencing—where manipulation can start at any time—we select 1,000 videos from Celeb-DF-v2 [27], one of the most challenging public deepfake datasets. It contains 590 real YouTube videos and 5,639 deepfakes generated from them. For each source, we interleave real and fake segments to produce streams alternating between authentic and manipulated frames (Figure 7b). This creates natural transitions, resembling real-time deepfake filters on webcam feeds (e.g., DeepFaceLive [19]). We refer to this as *Celeb-LiveDF*.

Key-frame-only detection. Figure 4 plots key-frames over time and encoded frame sizes when streaming *Celeb-LiveDF* as a webcam feed in a conference. Streams start at low resolution and ramp up; each resolution change triggers a key-frame, starting a new Group of Pictures (GOP). Crucially, when the real face is replaced by a deepfake face within a 1080p GOP, we observe *no additional key-frames*. Similarly, when a new deepfake face appears mid-stream, there is no key-frame being generated. These results suggest that modern deepfake generation methods introduce smooth changes that do not significantly disrupt the encoder. Consequently, a naive strategy that decodes and runs detection only on key-frames would miss such mid-stream injections.

Periodic detection. Key-frame-only detection can be easily bypassed, so a second baseline is to run detection periodically (every N frames). In practice, this still requires decoding all frames: key-frames are infrequent, and inter-frame dependencies prevent selective decoding. Detection takes 10 ms, which periodic execution can partially hide via buffering and overlap with decoding. However, as discussed in §3.1 and shown in Figure 3a, per-stream decoding does not scale. With inference every 30 frames, *Jitsi+DF* sustains only 80 conferences with quality dropping from 720p to 360p.

Table 1: Average Per-Operation Latency (μs) Comparison.

System	Frame Trans.	Detection Result Fetch	Packet Encrypt
Jitsi+DF	32.7	12.3	3.4
DeepSFU	5.2	9.7	0.9

Table 2: Media Packet Processing CPU Time Breakdown ($n = 15$ participants; decryption $\propto n$, others $\propto n^2$).

	Encrypt	Replicate	Bitrate Control	I/O	Decrypt
%CPU	49.97	15.21	14.78	14.28	5.75

3.3 Opportunities for Acceleration

Video-based acceleration. Modern codecs exploit temporal dependencies for efficient compression: natural facial motion evolves smoothly, while live deepfake injection introduces abrupt changes that disrupt continuity. This creates an opportunity to use lightweight cues present in encoded frames to infer suspicious changes.

- **Spatial and temporal encoded features.** Encoded frames reflect how content changes over time and where changes concentrate. When changes are smooth and localized, compression behavior remains stable; when abrupt and broad, the encoded stream shifts sharply. These spatial and temporal signals serve as indicators of potential deepfake injection.
- **Key-frame fallback.** Instead of continuous decoding, one can decode only when encoded stream suggests suspicious transition. Then, requesting a key-frame provides a clean reference point for inference while maintaining efficiency.

HW-based acceleration. SmartNICs offer a clear opportunity to free host CPU by offloading routine packet processing tasks. This eliminates contention between media-forwarding and deepfake-detection paths, thereby preserving QoE.

- **Encoded-frame GPUDirect RDMA.** SmartNIC ARM cores can execute lightweight logic effectively and, when paired with GPUDirect RDMA [39], can assemble encoded frames and transfer them directly into GPU memory. This bypasses host staging, lowering host CPU overhead and transfer latency (Table 1).
- **Crypto-hardware offloading.** Modern SmartNICs expose hardware cryptographic engines that can offload per-receiver HBH encryption (§2.2). This is especially impactful because encryption workload dominates media packet processing cost (Table 2).

4 *DeepSFU* Design

In this section, we outline three key design considerations, followed by how we realize *DeepSFU*. Figure 5 provides *DeepSFU*'s design overview, illustrating the division of labor between the media-forwarding and deepfake-detection pipelines.

4.1 Design Considerations

Deepfake detection as an infrastructure service. In principle, deepfake detection could run on clients, since endpoints already decode video for rendering. In practice, accurate detectors are heavy DNNs that benefit from discrete GPUs, which many commodity laptops and phones lack; moreover, on-device inference adds latency, heat, battery drain, degrading QoE. Offering detection as an infrastructure service lowers the endpoint burden by shifting compute and maintenance off clients while providing transparent protection as synthetic media proliferates. Accordingly, *DeepSFU* performs

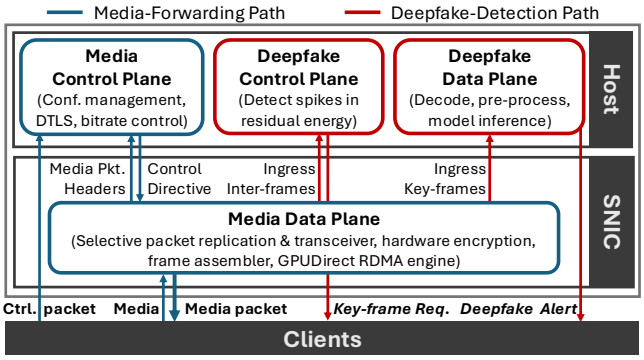


Figure 5: *DeepSFU* design overview.

detection at the SFU while addressing the scalability bottlenecks in §3.1, requiring minimal SFU changes and no client modifications.

Balancing SFU resources for forwarding and detection. Performing deepfake detection at the SFU introduces non-trivial contention between the media-forwarding and detection pipelines. In particular, *(i)* detection requires frequent host-to-GPU frame transfers that consume substantial CPU cycles, and *(ii)* media forwarding incurs heavy per-receiver packet encryption on the CPU. These demands compete for SFU resources, reducing throughput and bitrate, as shown in Figures 3b and 3c. To preserve media quality, *DeepSFU* must therefore orchestrate resource usage across the pipelines.

Decoding bottlenecks in deepfake detection. Even with *Jitsi+DF* highly optimized pipeline, which keeps decoding, pre-processing, and inference entirely on the GPU to minimize data movement, scalability remains severely limited, because high-accuracy detection requires high-resolution pixel frames. This forces the SFU to decode every stream centrally, placing heavy pressure on both the CPU and the GPU decoder (§3.1). As load increases, per-frame processing latency quickly exceeds the inter-frame budget, leading to frame drops and decoder failures. To scale, *DeepSFU* must therefore avoid continuous per-stream decoding while still delivering accurate, low-latency deepfake detection.

4.2 Split Forwarding and Deepfake Detection

The contention between media forwarding and deepfake detection is a key scalable bottleneck (§3). Therefore, *DeepSFU* carefully orchestrates resource usage to scale deepfake detection while preserving high SFU throughput (Figure 5).

Media-forwarding pipeline. As shown in Table 2, complex bitrate-control logic accounts for only ~15% of CPU time, whereas per-packet primitives dominate CPU consumption. *DeepSFU* therefore offloads these primitives to the SmartNIC, freeing host CPU cycles for higher-value deepfake detection tasks. Because the bitrate controller can estimate each ingress stream's rate from packet headers alone, *DeepSFU* separates the forwarding pipeline into a SmartNIC fast path (data plane) and a host slow path (control plane).

- **Media control plane (host CPU).** It runs complex, stateful logic such as conference management (join/leave), DTLS, and bitrate control, and processes control traffic carrying client feedback (e.g., available bandwidth and loss rates). It also provides the data plane with the metadata needed for forwarding (e.g., encryption keys and ingress to egress stream mappings) and media-quality decisions (e.g., target resolution and frame rates).

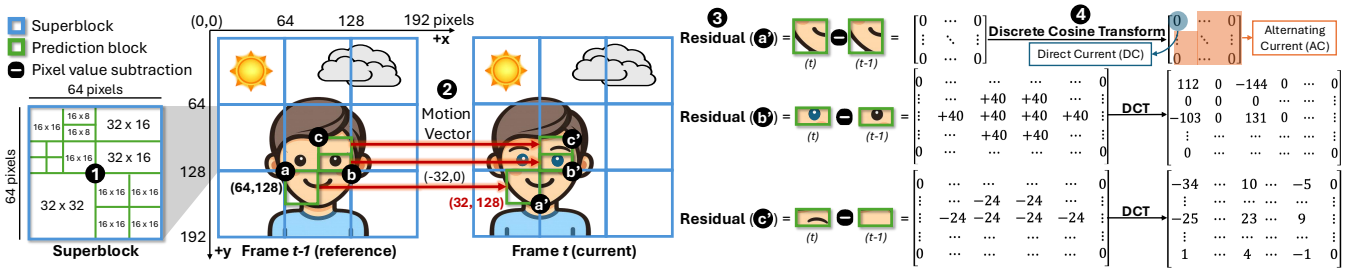


Figure 6: VP9 inter-frame encoding. The encoder predicts the current frame from a reference frame, ❶ partitions it into 64×64 superblocks and smaller prediction blocks, ❷ searches the ref. frame for the best-matching region and signals a motion vector for each prediction block, ❸ forms a residual (cur−ref), and ❹ transforms it into DC (average) and AC (detail) coefficients.

- **Media data plane (SmartNIC).** For each ingress media packet, it decrypts and parses headers, then forwards lightweight header metadata to the control plane for ingress-rate estimation and quality decisions. With the resulting per-receiver quality selection, it replicates only packets from the chosen quality and re-encrypts them per receiver.
- **Pipelined crypto (SmartNIC).** HBH encryption makes the SFU's cryptographic workload inherently asymmetric: each ingress packet is decrypted once, then replicated and re-encrypted for each receiver (§2.2). *DeepSFU* exploits the SmartNIC's hardware crypto engine with a pipeline where per-receiver encryption runs asynchronously as packets are decrypted and transmitted.

Deepfake-detection pipeline. Per-stream decoding dominates latency because inference requires pixel frames. *DeepSFU* therefore leverages lightweight yet informative spatial and temporal signals in encoded frames to guide *selective* decoding and model inference.

- **Deepfake control plane (host CPU).** Exploiting dependencies across consecutive frames, the control plane extracts/parses and quantifies changes in the facial region directly from encoded data, avoiding expensive per-frame decoding. Upon detecting abrupt facial-region changes, it issues a key-frame request, prompting the encoder to emit a key-frame as soon as possible.
- **Deepfake data plane (GPU).** When a key-frame arrives, the media data plane transfers it to GPU memory for decoding, preprocessing, and inference. If a deepfake is detected, *DeepSFU* alerts clients. By decoding only on-demand key-frames, *DeepSFU* avoids per-frame per-stream decoding and reduces GPU-decoder contention as conferences scale.

Why not SmartNIC-only detection? Since the SmartNIC media data plane already reassembles encoded frames, one natural alternative is to run deepfake detection directly on the SmartNIC itself without transferring frames to host CPU. *DeepSFU* intentionally avoids this placement to preserve resource separation: although the deepfake-onset detection (§4.3) avoids expensive per-frame decoding, it still imposes substantial compute overhead that contends with the media-forwarding path when co-located on the SmartNIC ARM cores. This SmartNIC-specific challenge is quantified in §7.8 with a *naive baseline* that runs the onset detection on all 16 BF-3 ARM cores; it scales to only 303 concurrent conferences ("V5" in Table 7), while a similar variant with host-side detection ("*DeepSFU* w/o Pipeline & RDMA") supports 2.4× more conferences and full *DeepSFU* reaches 4.3×. This supports our split design for media-forwarding and deepfake-detection pipelines across host/SmartNIC.

Efficient forwarding-to-detection data path. Repeated CPU–GPU frame transfers incur high CPU overhead (Figure 3b). *DeepSFU* eliminates this cost by designing a zero-copy frame data path that uses GPUDirect RDMA between the SmartNIC and the GPU.

4.3 Tracking Facial Changes Without Decoding

Fully decoding every frame does not scale, as it imposes substantial load on the SFU. We therefore leverage how modern codecs encode inter-frame changes and propose a decode-free method to track facial-region manipulations, which significantly reduces how often *DeepSFU* must decode and run deepfake inference. Figure 6 illustrates how VP9 (the codec commonly used in SFU) encodes inter-frames (1)–(4); our approach uses residual energy that quantifies the magnitude of changes in facial regions.

(1) Hierarchical block partitioning. Motion-compensated video encoders (e.g., H.265, AV1, VP9) compress raw frames into compact bitstreams by exploiting temporal redundancy, i.e., encoding each frame largely as changes relative to a prior *reference frame*. To efficiently represent these inter-frame changes, the encoder first partitions the current frame into a hierarchy of blocks: it tiles the frame into *superblocks* (64×64 pixels) and then adaptively subdivides each superblock into smaller *prediction blocks* (e.g., 32×32, 16×16, etc.). This adaptive partitioning matches scene complexity: smooth or static regions (e.g., uniform backgrounds) are well modeled with larger blocks, while detailed or highly dynamic regions (e.g., facial expressions) benefit from finer partitions.

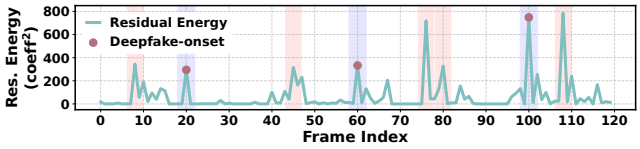
(2) Motion estimation. The encoder then searches the reference frame for each prediction block's best-matching region. The offset between the block's position in the current frame and its match in the reference is encoded as a *motion vector*, capturing apparent pixel movement so the encoder can reuse reference content instead of re-encoding it. For example, a' in the current frame is best matched by a in the reference, and the encoder signals the motion vector:

$$(\Delta x, \Delta y) = (32 - 64, 128 - 128) = (-32, 0) \text{ pixels.}$$

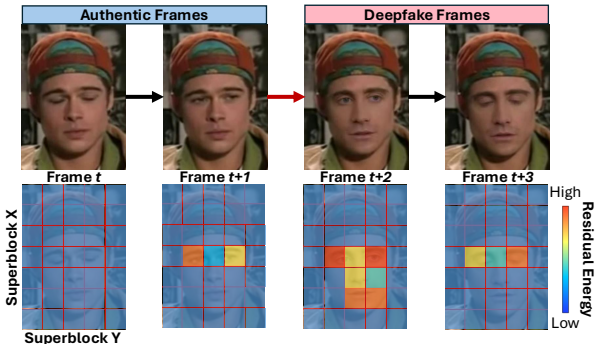
(3) Residual computation. When the matched regions are highly similar, motion compensation reuses most pixels, leaving a small *residual* to encode. In contrast, rapid appearance changes (b' & b) or new details (c' & c) increase it. Intuitively, residual captures the "new" information that cannot be explained by motion:

$$R(x, y) = C(x, y) - \hat{C}(x, y),$$

where $C(x, y)$ denotes the pixel value in the current prediction block and $\hat{C}(x, y)$ is its counterpart from the reference.



(a) Average residual energy over time. Deepfake-onset frames produce pronounced local spikes (blue regions), but normal facial expressions may also produce similar spikes (red regions).



(b) The onset frame (*t+2*) concentrates residual energy over a broader cluster of facial superblocks than expression-driven changes.

Figure 7: *DeepSFU*'s complementary deepfake-onset indicators: (a) residual energy (temporal); (b) cluster score (spatial).

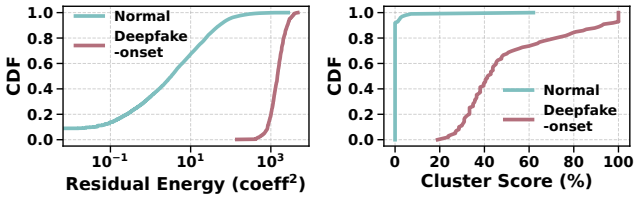
(4) Discrete cosine transformation. The residual still lives in the spatial domain, so encoding it raw would demand substantial data. To compact it, VP9 applies the *discrete cosine transform* (DCT), mapping the residual into the frequency domain where energy concentrates in a few coefficients. The resulting matrix has a *Direct Current* (DC) term at index $(0, 0)$, capturing average color/brightness changes, and *Alternating Current* (AC) terms capturing higher-frequency variation (edges, gradients). An eye-color change (b²) yields a strong DC response, while a newly appearing eyebrow (c²) spreads energy across many AC coefficients.

Residual energy for change quantification. To quantify facial changes with minimal computational overhead, we compute *residual energy* by aggregating frequency-domain coefficients. Let $b \in \mathcal{B}$ index the set of prediction blocks contained in a superblock, and let $C_b(u, v)$ denote the DCT coefficient at index (u, v) of block b, whose width and height are W_b and H_b. We define the superblock residual energy as:

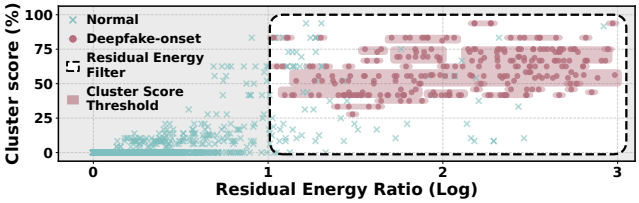
$$E_{\text{SB}} = \sum_{b \in \mathcal{B}} \sum_{u=0}^{W_b-1} \sum_{v=0}^{H_b-1} C_b(u, v)^2.$$

Intuitively, a larger E_{SB} indicates a greater mismatch between the current and reference content within the superblock, because the encoder must use more (or higher-magnitude) coefficients to represent the changes. Based on the analysis of deepfake video streams, we observe two distinctive residual-energy characteristics that inform *DeepSFU*'s complementary deepfake-onset indicators:

- **Average residual energy.** Figure 7a shows the average residual energy across facial superblocks, capturing the temporal magnitude of a facial change. As highlighted by the blue regions, deepfake-onset frames form pronounced local maxima, which are detectable through a simple statistical filter. However, normal facial expressions (red regions) produce similar spikes, making average residual energy ambiguous as a standalone indicator.



(a) Onset frames show markedly higher residual energy and cluster scores.



(b) Two-stage detector in action: *(i)* residual-energy filtering followed by *(ii)* adaptive cluster-score thresholding isolates deepfake-onset frames accurately.

Figure 8: *DeepSFU*'s deepfake-onset detection rationale.

- **Cluster score.** As illustrated in Figure 7b, deepfake-onset frames exhibit high residual energy across a broader region of the face (Frame *t+2*), whereas normal expression-driven changes tend to remain localized (Frame *t+1*). This spatial extent can be captured by measuring the fraction of superblocks whose energy is relatively higher than that of recent frames. We leverage this cluster score as the second indicator to further isolate onset frames.

Two-stage deepfake-onset detection. Figure 8a plots the cumulative distributions of both indicators on the *Celeb-LiveDF* dataset. Deepfake-onset frames shift the curves rightward, reflecting substantially higher residual energy and stronger clustering. However, the remaining overlap with normal frames makes a single fixed detection boundary unreliable, while absolute residual-energy statistics can also vary with video content and encoding conditions. Moreover, machine-learning (ML) approaches require offline training and add computational overhead, making them ill-suited for real-time deployment. *DeepSFU* therefore adopts a lightweight, causal two-stage onset detector.

Figure 8b plots per-frame cluster scores versus residual energy ratios for illustration, where the energy ratio is presented as the log-scaled quotient of the average residual energy and its threshold ($\bar{E}_f$ and $E_{threshold}$ in Algorithm 2). For *DeepSFU*'s causal deepfake-onset detection, Stage 1 applies a residual-energy filter that selects candidate onset frames with high recall. Stage 2 then applies an adaptive cluster-score threshold to reject false positives among the candidates with high but spatially localized residual energy. Both stages rely solely on prior frames, allowing the detector to adapt to each video stream online without supervised training. §5.4 and Appendix A detail these statistical stages. Each detected onset event is used to trigger a key-frame request for timely authenticity verification through decoding and detection model (SBI [50]) inference.

Table 3: Deepfake-Onset Detection Method Comparison.

Method	Precision	Recall	F1-score	Accuracy	PR-AUC
Logistic Regression	0.675	0.992	0.803	0.990	0.833
Random Forest	0.680	0.995	0.808	0.990	0.838
Support Vector Machine	0.775	0.995	0.871	0.994	0.885
(Stage 1) Energy-Filter Only	0.724	0.987	0.835	0.992	0.855
(Stage 2) Cluster-Score Threshold Only	0.818	0.982	0.892	0.995	0.900
Proposed Two-Stage Detection	0.958	0.976	0.967	0.999	0.967

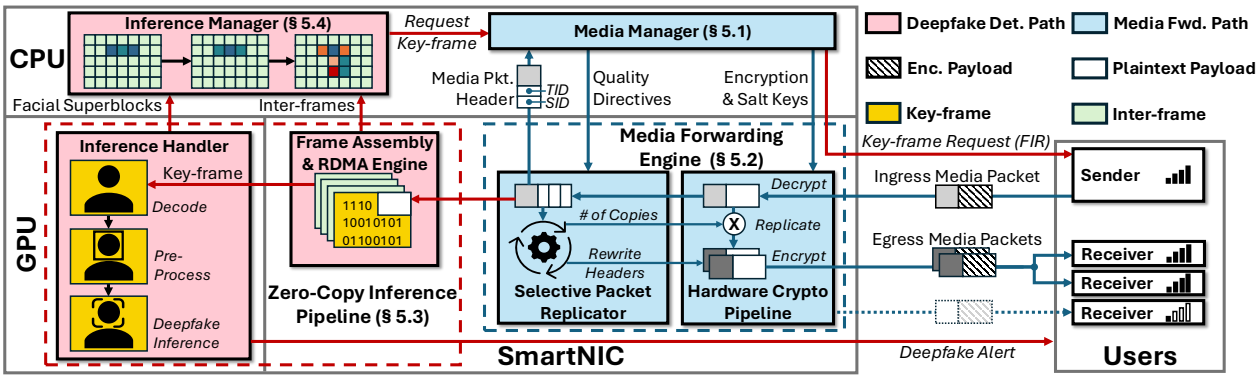


Figure 9: *DeepSFU* system architecture. Processing paths: *(1) media forwarding*—packet processing on the SmartNIC with per-receiver quality control; and *(2) deepfake detection*—frames are reassembled and RDMA-ed to the host CPU (inter-frames, ~99%, for decode-free onset detection) or the GPU (key-frames, ~1%, for decoding and inference).

Design validation. Table 3 compares our two-stage onset detector with its individual single-indicator stages and representative ML baselines. Although the ML-based methods achieve high recall, feature overlap between normal and deepfake-onset frames produces more false positives and the resulting lower precision despite the use of trained classifiers. On the other hand, our residual-energy filter (Stage 1) is intentionally designed as a recall-oriented step, identifying deepfake-onset candidates with few missed detections, at the cost of additional false positives. The cluster-score thresholding (Stage 2) then refines these candidates by filtering out normal high-energy frames that are frequently mistaken for deepfake-onset frames, substantially reducing false positives while preserving high recall. By combining both stages, *DeepSFU*'s causal onset detector achieves the best precision, F1-score, accuracy, and PR-AUC at moderate recall, without requiring labeled data or model training.

5 *DeepSFU* System

We introduce *DeepSFU*, a system that scales deepfake detection across many conferences. Figure 9 presents the *DeepSFU*'s system architecture. Ingress traffic follows two parallel processing paths: *media forwarding* and *deepfake detection*.

5.1 Media Manager

To process media traffic entirely on the SmartNIC and to issue key-frame requests, the Media Manager uses a small set of control APIs to interact with the Inference Manager, the Selective Packet Replicator (SPR), and the Hardware Crypto Pipeline (HCP):

- reg_keys(); After DTLS handshake, the Media Manager invokes this API to register the encryption and salt keys with the HCP.
- bind_streams(); Since SPR maintains only per-stream state, the Media Manager calls this API to bind an ingress to per-receiver egress stream specifying replication relationships.
- set_layers(); Media Manager uses this API to configure the target SVC spatial (SID) and temporal (TID) layer indices for each receiver, selecting suitable resolution and framerate.
- request_keyframe(); On a deepfake-onset detection, Inference Manager invokes this API to have Media Manager send a key-frame request (a Full Intra Request SRTCP packet), prompting the encoder to emit a key-frame for timely authenticity verification.

5.2 Media Forwarding Engine

Crypto trade-off and constraints. HBH encryption creates a dominant and asymmetric crypto workload (Table 2): each ingress packet is decrypted once, but every per-receiver replica must be encrypted with a different key. Thus, the key design trade-off is whether to keep this repeated egress encryption on the ARM cores (software), or to offload it to the SmartNIC crypto engine (hardware) despite an extra copy overhead. Specifically, *DeepSFU*'s design is shaped by the following constraints.

- **Per-receiver authentication tag.** In modern crypto suites, the authentication tag covers both the RTP header (as AAD) and the encrypted payload. Since each receiver's RTP header is unique, both paths cannot reuse one plaintext buffer across receivers.
- **Crypto engine buffers.** The crypto engine operates only on pre-allocated source and destination buffers registered during initialization as shown in Figure 10. The hardware path therefore cannot encrypt packets in-place.
- **Asynchronous NIC transmission.** NIC datapath is asynchronous with no completion callback. Hardware path therefore must copy ciphertext into a dedicated egress buffer upon encryption completion, so the engine's buffers can be reused immediately.

Hardware crypto pipeline. Figure 10 compares the two design options. An ingress packet ❶ arrives at the NIC and is DMA-ed into a ingress mbuf, where the ARM core ❷ decrypts it in place. On the software path, the ARM core ❸' replicates the decrypted packet directly into the egress mbuf and ❹' encrypts the replica in place; this avoids an extra replication but serializes per-receiver encryption on the ARM core. On the hardware path, the ARM core

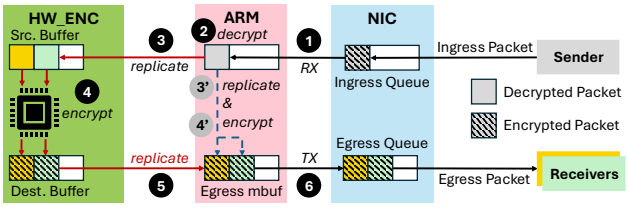


Figure 10: Hardware and software encryption comparison. Despite an extra copy (❺), hardware crypto offloading (❸–❺) is faster than encrypting in place on the ARM core (❸'–❹').

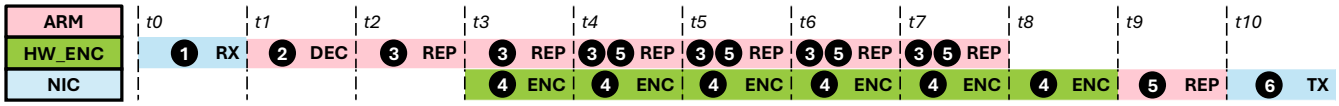


Figure 11: Hardware Crypto Pipeline. At *t4–t7*, while the crypto engine ❹ encrypts the current replica, the ARM core ❸ stages next receiver's copy into the source buffer and ❺ copies the ciphertext out—hiding both copies under encryption. Egress packets are ❻ transmitted in batches.

instead ❸ copies a per-receiver replica into the crypto engine's source buffer for ❹ hardware encryption, after which the ciphertext is ❺ replicated into an egress mbuf, ❻ DMA-ed to the NIC, and transmitted to the receiver. *DeepSFU* adopts this hardware path: one extra copy (❺) plus hardware encryption costs only 528 ns per packet, 2.7× faster than software encryption, and offloading frees the ARM cores for SPR and other detection tasks. To further mask this copy overhead, we build a pipelined design that overlaps work across the ARM cores, crypto engine, and NIC datapath (Figure 11).

Selective packet replicator. To remain compatible with endpoint devices, *DeepSFU* supports both bitrate adaptation modes: simulcast and SVC. Because media traffic is processed entirely on the SmartNIC, SPR is responsible for enforcing per-receiver quality decisions at packet level.

- **Simulcast.** Each sender encodes multiple qualities as independent streams. According to mapping from bind_streams(), SPR rewrites each ingress packet's header (SSRC, sequence number, timestamp) to its egress values while preserving inter-frame pacing. Simulcast adaptation is achieved by remapping an egress to a different ingress stream.
- **SVC.** Building on simulcast, SVC is more complex because multiple quality layers are multiplexed within a single ingress stream. SPR parses the VP9 payload descriptor to extract SID/TID, then uses per-receiver targets from set_layers() to filter packets: receivers requesting lower frame rate or resolution drop higher TIDs or SIDs, respectively. SPR then informs HCP how many copies each ingress packet needs.

5.3 Zero-Copy Inference Pipeline

As shown in §3.1, frame transfer is the dominant bottleneck and a major source of CPU contention. *DeepSFU* therefore depacketizes each ingress stream on the SmartNIC into complete encoded frames and RDMA-writes them directly into CPU or GPU pinned memory: over 99% of frames are inter-frames, which the Inference Manager (§5.4) analyzes using lightweight deepfake-onset detection, while the remaining ~1% key-frames are written to GPU memory for the Inference Handler to decode and run model inference.

Frame assembly & RDMA engine. From each RTP packet, *DeepSFU* extracts the timestamp, sequence number, and B/M (begin/end-of-frame) marker bits. All packets in a frame share the same timestamp, so *DeepSFU* maintains a per-stream buffer keyed by it:

(1) A packet with B=1 opens a new frame.
(2) Packets with the same timestamp are appended and reordered by sequence number to absorb out-of-order arrivals.
(3) A packet with M=1 completes a frame; the assembled bitstream is written to GPU/CPU memory using the RDMA engine.

Two fallbacks handle packet loss: *(i)* if the M packet is lost or a sequence number is still missing when a new timestamp arrives, the pending buffer is retained while the new frame assembles in a separate buffer; *(ii)* any buffer idle for more than a certain amount of time (e.g., 1s) is timed out and flushed to prevent stale state.

Inference Handler. When a key-frame request is triggered, the Inference Handler processes the resulting key-frame over a three-stage pipeline: *(i)* frame decoding, *(ii)* frame pre-processing, and *(iii)* model inference. Since the pipeline runs infrequently (on only a few frames per stream), its per-frame latency remains negligible.

- **Frame decoding.** RDMA-ed key-frames are decoded using GPU decoder, producing frames directly in GPU memory. Since only key-frames are decoded, no inter-frame state or ordering is required, making the process resilient to transient frame loss.
- **Frame pre-processing.** Pre-processing is performed entirely on the GPU, including color-space conversion and spatial transformations required by the model.
- **Detection model inference.** Inference consists of two steps: *(i)* face detection and *(ii)* deepfake detection, both performed independently per key-frame with no temporal state across frames. This enables the Inference Handler to pipeline both steps to concurrently serve frames across and within streams. Upon completion, the classification result is returned directly to the client.

5.4 Inference Manager

The Inference Manager tracks facial changes decode-free by *(i)* parsing inter-frame transform coefficients and computing average residual energy over facial superblocks, then *(ii)* detecting deepfake onset using a lightweight statistical method over the two indicators described in §4.3. Upon detection, the Inference Manager invokes request_keyframe() to send a FIR as an SRTCP packet.

Parsing facial superblock residual energy. The VP9 decoding pipeline has four stages: *(1)* bitstream entropy parsing to extract quantized coefficients, *(2)* dequantization to restore transform-domain values, *(3)* inverse DCT to convert coefficients from frequency to spatial domain, *(4)* prediction addition and loop filtering. For inter-frames, we tap the pipeline after stage *(1)* to extract quantized transform coefficients, avoiding expensive computation. Algorithm 1 summarizes the workflow. After entropy decoding at superblock level, residual energy is computed for face-region superblocks by aggregating squared DC and AC coefficients across transform blocks. These per-frame residual energies are fed into Algorithm 2 for real-time deepfake-onset detection.

Deepfake-onset detection. Building on the two onset indicators as discussed in §4.3, our detector avoids heavy computation and instead uses lightweight, real-time statistical tests. Algorithm 2 summarizes the overall procedure.

- **Stage 1: Localized non-parametric statistical test.** Under general encoding environments, residual energy values do not follow a fixed normal distribution. To address this, we build a real-time 'normal state' distribution based on data from the previous W (e.g., 30) frames. When the mean residual energy of the

Algorithm 1 Inter-frame Residual Energy Parsing

Input: Encoded inter-frame f // From RDMA Engine.
Set of facial superblocks $\mathcal{F}_{sb}$ // From Inference Handler.
Output: Facial-superblock residual-energy array $\mathbf{E}_f$

procedure FrameParser($f, \mathcal{F}_{sb}$):
  $\mathbf{E}_f \leftarrow [\,]$ // Initialize energy array for frame
  for each superblock sb in $\mathcal{F}_{sb}$ **do**
    $DC \leftarrow 0, AC \leftarrow 0, Q \leftarrow 0$ // Initialize variables
    // Aggregate transform block coeff.
    for each transform block tb in sb **do**
      $Q \leftarrow$ ParseArithmeticCodes(tb)
      $DC \leftarrow DC + (Q[0])^2$ // Aggregated DC coeff.
      $AC \leftarrow AC + \sum_{i=1}^{n}(Q[i])^2$ // Aggregated AC coeff.
    $E \leftarrow DC + AC$ // Total energy of the superblock
    $\mathbf{E}_f$.Append(E) // Append tot.energy to the result
  return $\mathbf{E}_f$

Algorithm 2 Real-time Deepfake-onset Detection

Global: Window size $(W) \leftarrow 30$, Sampling size $(N) \leftarrow 1000$,
Significance level $(\alpha) \leftarrow 0.05$, Weight factor $(w) \leftarrow 1.5$
Maintained state: Avg.energy sliding window $\{\bar{E}_{f-W}, \ldots, \bar{E}_{f-1}\}$,
Avg. cluster score sliding window $\{\bar{C}_{f-W}, \ldots, \bar{C}_{f-1}\}$
Input: Facial-superblock residual-energy array E_f
Output: *Normal* or *Deepfake-onset* flag

procedure OnsetDetector($\mathbf{E}_f$):
  // Stage 1: Localized Statistical Test
  $\bar{E}_f \leftarrow$ Mean($\mathbf{E}_f$) // Average facial residual energy
  $\mathcal{D} \leftarrow$ Sample N values from $\{\bar{E}_{f-W}, \ldots, \bar{E}_{f-1}\}$
  Sort $\mathcal{D}$ in ascending order
  $E_{threshold} \leftarrow \mathcal{D}[\lceil N(1-\alpha) \rceil]$ // Top 5% threshold
  $E_{max} \leftarrow$ Max($\bar{E}_{f-W}, \ldots, \bar{E}_{f-1}$) // Sliding-window max
  if $\bar{E}_f \leq E_{threshold}$ **or** $\bar{E}_f < E_{max}$ **then**
    return *Normal* flag // Natural facial changes
  // Stage 2: Adaptive Cluster Score Filtering
  $\bar{C}_f \leftarrow$ %superblocks in $\mathbf{E}_f$ with $E > E_{threshold}$
  $T_{base} \leftarrow$ Mean($\bar{C}_{f-W}, \ldots, \bar{C}_{f-1}$) // Base threshold
  $\sigma_f \leftarrow$ Std($\bar{C}_{f-W}, \ldots, \bar{C}_{f-1}$) // Standard deviation
  $T_f \leftarrow T_{base} + w \cdot \sigma_f$ // Adaptive threshold
  if $\bar{C}_f \geq T_f$ **then**
    return *Deepfake-onset* flag // Send key-frame request
  else
    return *Normal* flag // Natural facial changes

current frame ($\bar{E}_f$) exceeds the statistical threshold ($E_{threshold}$) corresponding to the top 5% of this distribution while simultaneously representing a local maximum (E_{max}) within its temporal neighborhood, we identify it as a potential onset.

- **Stage 2: Adaptive cluster-score threshold.** Motivated by the spatially widespread facial changes at deepfake onset (Figure 7b), we validate Stage 1 candidates by measuring the spatial extent of anomalies within the face region. We quantify this effect using the *cluster score* $\bar{C}_f$, which captures the fraction of facial superblocks deemed anomalous. To adapt across videos and time-varying motion, we adjust the decision threshold T_f online based on recent-frame statistics (e.g., historical variance). Only candidates that exceed this adaptive criterion are labeled as deepfake-onset frames. Appendix A provides the formal statistical definitions and a step-by-step procedure.

6 Implementation

Media Controller (host). *DeepSFU*'s media controller extends JVB [21] with stream-management and full-intra request support. Importantly, these additions require only modest changes to the SFU codebase—388 lines modified out of 69,409 total—while all other Jitsi Meet components [22, 23] remain unmodified.

Inference Handler & Manager (host). The Inference Handler is a C++ program that uses the NVIDIA Video Codec SDK [40] to feed frames into the GPU decoder and convert them from NV12 to RGB via a custom CUDA kernel. We adopt the SelfBlended-Images [50] model, converted to ONNX and integrated into the Inference Handler. The Inference Manager is a C++ program that extends libvpx [55] to export quantized DC/AC coefficients.

SmartNIC Stack. *DeepSFU*'s SmartNIC stack is implemented as a DPDK [45] application running on the BlueField-3 [37] platform. We configure Open vSwitch [46] and Scalable Functions [36] to steer ingress traffic through the ARM core. *DeepSFU* decrypts packets using OpenSSL [41] and offloads packet encryption to the BlueField-3 crypto engine via the DOCA AES-GCM library (AES-GCM-128) [38]. For frame movement, *DeepSFU* uses GPUDirect RDMA [39]. When the destination is host memory, it uses RDMA Verbs [28].

7 Evaluation

In this section, we evaluate the performance of *DeepSFU* against *Jitsi+DF*, guided by the following questions. **Q1:** How many more concurrent conferences can *DeepSFU* support with continuous detection? Refer to §7.2. **Q2:** How much does *DeepSFU* reduce per-frame latency? Refer to §7.3. **Q3:** How accurate is *DeepSFU* at detecting deepfake onset? Refer to §7.4. **Q4:** How much faster does *DeepSFU* detect and alert deepfakes? Refer to §7.5. **Q5:** What bandwidth overhead does *DeepSFU* add, and what throughput gain does it enable? Refer to §7.6. **Q6:** How does the choice of deepfake detection model affect the scalability of *DeepSFU* and *Jitsi+DF*? Refer to §7.7. **Q7:** How much does each *DeepSFU* component contribute to scalability? Refer to §7.8.

7.1 Experiment Setup

Our experiments run on a cluster with 100 bare-metal machines acting as load-testing clients and a dedicated workstation hosting the SFU. All nodes are connected through a Top-of-Rack L2 switch (100 Gbps). Table 4a summarizes machine specifications. To ensure fairness, we deploy *DeepSFU* and *Jitsi+DF* on identical infrastructure. For *Jitsi+DF*, we configure the BlueField-3 in NIC[1] mode so packets bypass the on-board ARM cores. The evaluation scenarios are presented in Table 4b using typical conference size of 15 users.

Dataset coverage. As mentioned in §3.2, *Celeb-LiveDF* is built by interleaving frames from the Celeb-DF-v2 [27] dataset, whose videos are drawn from publicly available YouTube sources featuring 59 celebrities. As summarized in Table 4c, *Celeb-LiveDF* inherits this breadth of subjects and visual conditions, spanning a wide range of demographics, brightness levels, head motion, and face sizes. This makes it ideal for evaluating the robustness of *DeepSFU*'s deepfake-onset detection against the variability one would expect in deepfake content encountered in the wild.

[1] In NIC mode, the BlueField-3 functions as a ConnectX-7 adapter.

Table 4: Evaluation Setup.

(a) Hardware Specifications.

Component	Client Machine	SFU Machine
CPU	Intel E5-2680 v2	AMD Ryzen 7 9800X3D
Memory	64GB DDR3	64GB DDR5
GPU	N/A	NVIDIA RTX PRO 6000
NIC	10G Mellanox ConnectX-3	400G NVIDIA BlueField-3

(b) Evaluation Scenarios.

Difficulty Level	Deepfake Injection Timing
Easy	Deepfake is present from session start
Medium	Deepfake is injected during the session
Hard	A new deepfake face appears mid-session

(c) *Celeb-LiveDF* Dataset Composition.

Attribute	Category	# Videos	%
Gender	Male, Female		
Age group	< 30, 30s, 40s, 50–60, ≥ 60		
Ethnicity	Asian, African American, Caucasian		
Brightness (mean luma, 0–255)	Low (< 50)	320	32.0
	Medium (50–90)	447	44.7
	High (> 90)	233	23.3
Head Motion (head-pose std, °)	Low (< 8)	339	33.9
	Medium (8–14)	422	42.2
	High (> 14)	239	23.9
Face Size (face / frame height)	Small (< 0.32)	263	26.2
	Medium (0.32–0.42)	484	48.4
	Big (> 0.42)	253	25.3

7.2 Scalability Improvements

Figure 12 compares the number of concurrent conferences *DeepSFU* and *Jitsi+DF* sustains while providing continuous deepfake detection and preserving high media quality, under the three increasingly challenging scenarios in Table 4b.

Scalability. *Jitsi+DF* scales well only in the easy case, where deepfakes appear at stream start: its *key-frame only* mode reaches 115 conferences (720p) because decoding and inference run only a few times per stream. But in medium and hard cases, mid-stream injections do not trigger key-frames (§3.2); per-frame decoding and periodic inference cut scalability to 50 conferences. In contrast, *DeepSFU* scales much better across all scenarios through lightweight, accurate deepfake-onset detection and media data plane offloading. In the hard case (a new participant joining an ongoing stream), *DeepSFU* adds only a simple DC-coefficient check on four boundary superblocks to detect possible person appearance and trigger key-frame verification, with minimal overhead (~5% scalability reduction). This advantage holds even at 1080p.

7.3 Latency Improvements

Per-frame latency improvement. Figure 13 compares the per-frame latency of *DeepSFU* and *Jitsi+DF* (full CPU provision). In *Jitsi+DF*, per-frame decoding causes latency to grow sharply as the number of streams increases, reaching the 33 ms inter-frame budget at only 90 conferences.

In contrast, *DeepSFU* performs only lightweight frame parsing, which is substantially cheaper at just 0.23 ms per frame. Notably, *DeepSFU*'s frame-transfer and result read latencies are negligible (18 μs). Frame pre-processing and model inference take 10.64 ms on average; because this step runs asynchronously, it does not need to complete within the inter-frame budget.

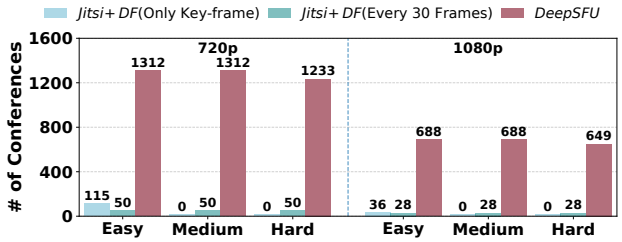


Figure 12: Scalability across the scenarios in Table 4b. *DeepSFU* sustains continuous detection and supports 26.2× (720p) and 24.6× (1080p) more conferences.

***DeepSFU*'s detection scalability.** To explain how *DeepSFU* sustains reliable detection at 688 (1080p) and 1,312 (720p) conferences, we measure the *aggregate* per-frame processing time of all ingress streams mapped to a single CPU core. Figure 14 shows that *DeepSFU*'s per-frame processing is so lightweight that one core can support up to 44 (1080p) or 82 (720p) conferences while staying within the inter-frame budget. This per-core capacity yields near-linear scaling with added cores by static partitioning: each ingress stream is pinned to a SmartNIC ARM worker and a host CPU core, and the ARM worker RDMAs frames into a core-private memory region. Because cores neither share packet buffers nor contend for DMA destinations, adding CPU/ARM cores increases supported conference count proportionally.

7.4 Deepfake-Onset Detection Accuracy

To verify the reliability of *DeepSFU*'s deepfake-onset detection, we evaluate it on 1,000 videos from *Celeb-LiveDF*. Each video contains 240 frames, including up to 5 *onset* frames where the content switches from real to deepfake. We count a detection as correct only if *DeepSFU* identifies the deepfake switching points precisely.

Confusion matrix. Table 5 reports the performance result. *DeepSFU* produces 209 false positives (FP) among 235,165 normal frames and 115 false negatives (FN) among 4,835 onset frames (Table 5a), showing the detector closely matches ground truth at scale.

Table 5: Deepfake-Onset Detection Results.

(a) Confusion Matrix.

	Predicted Onset	Predicted Normal
Actual Onset	4,720	115
Actual Normal	209	234,956

(b) Performance Metrics.

Metric	Prec.	Recall	F1	Spec.	Bal. Acc.	PR-AUC
Value	0.958	0.976	0.967	0.999	0.988	0.967

Performance metrics. As indicated in Table 5b, the model achieves high precision: 95.8% of frames flagged as *onset frames* are true onset events. It also attains excellent recall, correctly detecting 97.6% of all ground-truth onset frames. To account for the extreme class imbalance, we report balanced accuracy, which remains high at 98.8%, indicating strong performance on both normal and onset classes. Finally, the PR-AUC of 96.7% shows that the detector sustains a favorable precision–recall trade-off across decision thresholds. Overall, *DeepSFU* maintains low error rates and strong discriminative power. Appendix B further analyzes the False Negative cases.

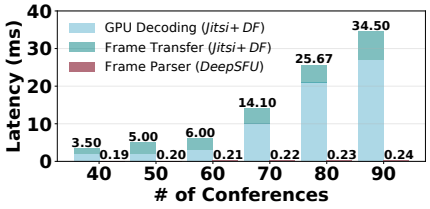


Figure 13: Per-frame latency comparison. *DeepSFU* lowers processing and frame-movement latency by 143.7×.

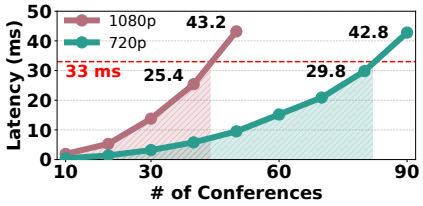


Figure 14: *DeepSFU* aggregated per-frame processing latency across all ingress streams on a single host CPU core.

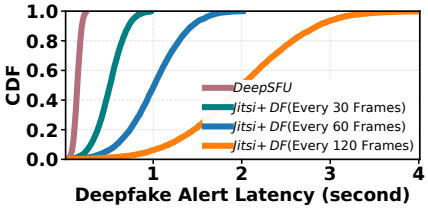
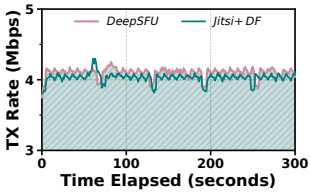
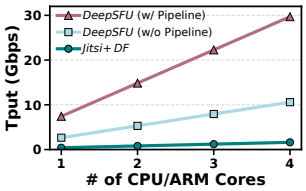


Figure 15: End-to-end deepfake alert latency comparison. *DeepSFU* raises alerts substantially faster than *Jitsi+DF*.



(a) Bandwidth usage.



(b) Aggregated throughput.

Figure 16: Bandwidth and throughput comparison. *DeepSFU* incurs only a modest bandwidth overhead (~0.74%), while delivering 18× higher throughput than *Jitsi+DF*.

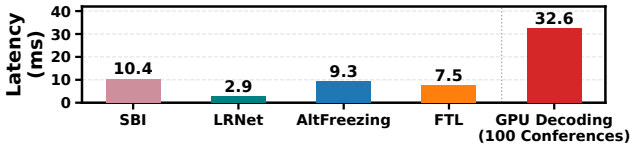


Figure 17: Per-frame decoding and inference latency across deepfake detection models. Regardless of model choice, the decoding overhead alone exhausts the 33 ms inter-frame budget at 100 concurrent conferences, imposing a hard scalability ceiling for *Jitsi+DF*.

7.5 Deepfake Detection and Alert Latency

To compare end-to-end deepfake alert latency for *DeepSFU* and *Jitsi+DF*, we measure the time from the first deepfaked video frame to when the corresponding alert is received at the receiver; Figure 15 plots the resulting CDF. *DeepSFU* performs deepfake-onset checks on every frame, allowing it to trigger a key-frame request immediately upon detecting a suspicious facial shift. The requested key-frame arrives only a few frames after the deepfake onset—typically 2–7 frames ($\mu = 4.3$, $\sigma = 1.2$)—yielding fast end-to-end alerting ($\mu = 153.1$ ms, $\sigma = 39.6$ ms). In contrast, *Jitsi+DF* fully decodes every frame and runs expensive model inference once every N frames; this reduced inference frequency introduces substantial delay before a deepfake can be reported.

7.6 Bandwidth and Throughput

Bandwidth overhead. To quantify *DeepSFU*'s bandwidth overhead, we measure bandwidth usage under identical video content and network conditions. Figure 16a reports the transmit bandwidth over a 5-minute conference. Because *DeepSFU*'s onset detection is highly accurate, it triggers only a few key-frame requests, resulting in modest ~0.74% overhead.

Table 6: Per-Packet Operation Latency Comparison (ns).

System	Encryption	Decryption	Replication	I/O
Jitsi+DF	3362	9622	1035	970
DeepSFU	349	1317	179	29

Throughput improvement. To quantify the benefits of *(i)* media data plane offloading and *(ii)* the hardware crypto pipeline, we compare the throughput of *Jitsi+DF* and *DeepSFU*. Figure 16b reports the aggregate throughput across all users: with a single ARM core running media data plane, *DeepSFU* supports 1,935 users across 129 conferences at ~7.4 Gbps, then scales nearly proportionally with additional ARM cores because the dominant media processing work is handled by hardware crypto. In contrast, *Jitsi+DF* exhibits limited scaling even when provisioned with more CPU resources. Table 6 further shows that *DeepSFU* reduces encryption latency by 9.6×; without hardware crypto pipelining, however, per-receiver encryption is serialized on the ARM cores, adding latency that degrades throughput and limits the benefit of media data plane offloading.

7.7 Impact of Deepfake Detection Models

Figure 17 shows that regardless of which detection model is used—SBI [50], LRNet [51], AltFreezing [60], or FTL [25]—the fundamental bottleneck lies upstream: *Jitsi+DF* must *decode* each video frame before passing it to the detection model. Note that media quality already degrades with 50 concurrent conferences (§3.1). *DeepSFU*, in contrast, performs decode-free onset detection and triggers decoding and inference on only ~1% of frames, so the model choice has no bearing on its scalability.

7.8 Ablation Study

We conduct an extensive ablation study that incrementally adds each system component of *DeepSFU* (and several combinations) to a common baseline. Table 7 reports the number of conferences with deepfake-detection enabled/disabled, and per-frame latency breakdown at a single conference load.

- ***Jitsi+DF.*** Media forwarding and frame transfer contend for the same CPU resources, limiting scalability (§3.1).
- **V1.** Adding onset detection improves scalability slightly. Although onset detection is lightweight, co-locating with media forwarding still quickly triggers bitrate degradation.
- **V2.** Adding RDMA, where the SmartNIC assembles and stages frames to GPU but does not forward media, frees the host CPU for media forwarding alone. Although media processing overhead appears low, scalability is still limited, reflecting the known scalability issue of software-based SFUs [30]. Meanwhile, detection remains capped at 100 by the decoding bottleneck (Figure 17).
- **V3.** Offloading the media data plane to SmartNIC *without* HW pipelining (3'–4' in Figure 10) substantially improves SFU scalability through control-/data-plane separation.

Table 7: Ablation Study. Overall scalability & per-frame latency breakdown at a single 15-user conference load.

	Variant	*DF Onset Detection*	*Zero Copy (RDMA)*	*Data plane Offload*	*HW Crypto Pipeline*	# of Concurrent Conferences			Per-Frame Latency (μs)			
						DF Det. ON	*DF Det.* OFF	Total	Media Proc.	Frame Trans.	GPU Decode	Onset Parser
w/o SNIC	***Jitsi+DF***	✗	✗	✗	✗	50	0	50	84.8	32.7	1,267	N/A
	V1	✓(Host)	✗	✗	✗	86	0	86	84.8	32.7	N/A	192
w/ SNIC	**V2**	✗	✓	✗	✗	100	15	115	84.8	5.2	1,267	N/A
	V3	✗	✗	✓	✗	100	636	736	22.6	32.7	1,267	N/A
	V4	✗	✗	✓	✓	100	1,994	2,094	7.9	32.7	1,267	N/A
	V5 (*naive SNIC baseline* in §4.2)	✓(SNIC)	✗	✓	✗	303	0	303	22.6	32.7	N/A	653
	***DeepSFU* w/o Pipeline & RDMA**	✓(Host)	✗	✓	✗	736	0	736	22.6	32.7	N/A	192
	***DeepSFU* w/o Pipeline**	✓(Host)	✓	✓	✗	736	0	736	22.6	5.2	N/A	192
	***DeepSFU* w/o RDMA**	✓(Host)	✗	✓	✓	1,248	846	2,094	7.9	32.7	N/A	192
	DeepSFU	✓(Host)	✓	✓	✓	1,312	782	2,094	7.9	5.2	N/A	192

- **V4.** Offloading the media data plane *with* HW pipelining (❸–❺ in Figure 10) yields a much larger SFU scalability gain, as the pipeline (Figure 11) hides encryption overhead.
- **V5.** As discussed in §4.2, this *naive SmartNIC baseline* shows that co-allocating onset detection and media data plane on SmartNIC causes ARM-core resource contention.
- ***DeepSFU* w/o Pipeline & RDMA.** Running onset detection on the powerful host CPU allows more deepfake-detection conferences. It also frees SmartNIC ARM cores for media data plane work, improving SFU scalability.
- ***DeepSFU* w/o Pipeline.** Although CPU-based onset detection can scale higher, media forwarding bottlenecks the system without HW pipelining, revealing its contribution compared to the full *DeepSFU*. Adding RDMA alone thus does not improve scalability, since its sole purpose is to relieve CPU of frame staging.
- ***DeepSFU* w/o RDMA.** HW pipelining removes the media forwarding bottleneck, raising total conferences, but without RDMA the host CPU must assemble and stage frames itself. This shows that CPU-based onset detection removes the dependence on GPUDirect RDMA, available only on server-grade GPUs (e.g., NVIDIA A100/H100). A comparison with V4 further isolates the contribution of onset detection, which serves as a key enabler for *DeepSFU* to scale substantially without expensive GPU feature.
- ***DeepSFU*.** Combining all four innovations, *DeepSFU* sustains a large number of conferences through HW pipelining. The deepfake-detection path is no longer bottlenecked by decoding and frame staging, raising deepfake-detection conferences to 1,312—the highest across all variants.

8 Discussion

Broader applicability of *DeepSFU*'s techniques. Two techniques are independent of deepfake detection: the hardware crypto pipeline (§5.2) and the zero-copy frame path (§5.3). They broadly apply to encrypted video analytics workloads where media streams must be decrypted, assembled, and delivered to accelerators. By avoiding host-side copies and crypto processing, these techniques free CPU cycles for media and model orchestration, inspired by the offload strategy behind recent ML inference serving systems [10].

Codec extensibility for *DeepSFU*'s onset detection. Our onset detection does not tie to a single codec. *DeepSFU* exports transform coefficients from libvpx [55] via a hook after entropy parsing; the same approach extends to FFmpeg-AV1 [14] and H.264 [7]. Once coefficients are exposed, the residual-energy computation (Algorithm 1) and onset detection (Algorithm 2) can be applied, as both operate on coefficients rather than codec-specific syntax.

Control- and data-plane separation. Separating a stateful control plane from a per-packet data plane is a well-established networking principle [18, 56] that *DeepSFU* adopts. We innovate upon this principle for the SFU's tightly coupled forwarding and detection pipelines: the media control plane and deepfake onset detection reside on the host, while per-packet primitives are offloaded to the SmartNIC, co-designed with the HW crypto pipeline and zero-copy frame path so both scale without contending for shared resources.

9 Related Work

Video conferencing. Recent video conferencing architecture (SFU) development grew alongside WebRTC adoption, supported by frameworks like MiroTalk SFU [8], Kurento [24], Licode [29], and Janus [20]. However, SFUs struggle with scalability during peak loads, particularly when replicating packets for multiple users [44]. Recent solutions have addressed this with P4 network switches [30].

Deepfake detection. Temporal methods like Temporal Coherence [65] and LRNet [51] exploit inter-frame consistency with lower compute but *(i)* create inter-frame dependencies, and *(ii)* sacrifice accuracy. Conversely, DNN detectors like SBI [50], RECCE [5], and Face X-Ray [26] achieve state-of-the-art per-frame accuracy. However, both categories require fully decoded frames.

10 Conclusion

In this work, we present *DeepSFU*, a system for real-time, scalable deepfake detection in video conferencing. *DeepSFU* breaks the fundamental tension between packet-level SFU scalability and frame-level deepfake detection by detecting deepfake onsets directly from encoded frames, avoiding fragile and costly per-frame decoding. *DeepSFU* also mitigates the SFU's asymmetric cryptographic workload with the hardware crypto pipeline. Our experiments show that *DeepSFU* reduces per-frame processing latency by 143.7× and supports accurate detection for 26.2× more concurrent conferences than the *Jitsi+DF* baseline, while preserving QoE.

Ethics: this work does not raise any ethical issues.

Acknowledgments

This work was supported by Cisco Systems under Grant No. 1368170, by the National Science Foundation under Grant Nos. 2326835 and 2241818, and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under Grant Nos. RS-2024-00405128 and 2026-RS-2020-II201819 funded by the Korea Government (MSIT). Sangtae Ha is the corresponding author.

References

[1] Bernard Aboba et al. 2015. Codec-Independent Selective Forwarding. Internet-Draft, IETF. https://datatracker.ietf.org/doc/html/draft-aboba-avtcore-sfu-rtp-00

[2] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. 2019. Protecting World Leaders Against Deep Fakes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[3] Bloomberg. 2024. How One Question Saved Ferrari From a Deepfake Scam. (2024). https://www.bloomberg.com/news/articles/2024-07-26/ferrari-narrowly-dodges-deepfake-scam-simulating-deal-hungry-ceo

[4] B. Burman, M. Westerlund, S. Nandakumar, and M. Zanaty. 2021. Using Simulcast in Session Description Protocol (SDP) and RTP Sessions. RFC 8853.

[5] Junyi Cao, Chao Ma, Taiping Yao, Shen Chen, Shouhong Ding, and Xiaokang Yang. 2022. End-to-End Reconstruction-Classification Learning for Face Forgery Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4113–4122.

[6] Channel NewsAsia. 2025. Company finance director nearly loses over US$499,000 to scammers using deepfake to impersonate CEO. (2025). https://www.channelnewsasia.com/singapore/deepfake-scam-impersonate-ceo-company-finance-director-5048706

[7] Cisco Systems. 2026. OpenH264: Open Source H.264 Codec. https://github.com/cisco/openh264. Accessed: 2026-06-20.

[8] MiroTalk Developers. 2025. MiroTalk SFU: Secure, Real-Time Video Conferencing. https://sfu.mirotalk.com/. Accessed: 2025-04-10.

[9] Dhanyalakshmi et al. 2026. Exploring the Advancements and Challenges of Deepfake Face-swap: A Survey. *Multimedia Tools and Applications* (Jan. 2026). https://link.springer.com/article/10.1007/s11042-026-21285-8 Accessed: 2026-06-06.

[10] Shirin Ebadi and Eric Keller. 2026. Optimizing Machine Learning Inference Serving Systems: A Survey. In *2026 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. 1–9. doi:10.1109/ACDSA67686.2026.11468130

[11] Entertainment Weekly. 2025. Fake videos of *General Hospital* star Steve Burton allegedly scam woman out of over US$80k. (2025). https://ew.com/fake-videos-of-general-hospital-star-steve-burton-scam-woman-out-of-life-savings-11800356

[12] Eurojust. 2025. Fraudulent call centres in Ukraine rolled up. Press release, European Union Agency for Criminal Justice Cooperation. https://www.eurojust.europa.eu/news/fraudulent-call-centres-ukraine-rolled Reports estimated damage over EUR 10 million to more than 400 known victims across Europe from impersonation-based fraud run via call centres in Ukraine..

[13] Nathanael Fast and Juliana Schroeder. 2023. Unveiling the Neely Ethics & Technology Indices. Psychology of Technology Institute (Substack); USC Marshall Neely Center Social Media Index. https://psychoftech.substack.com/p/unveiling-the-neely-ethics-and-technology Reports that 30.4% of U.S. adults used FaceTime in the past 28 days (nationally representative panel survey)..

[14] FFmpeg Developers. [n. d.]. FFmpeg. https://www.ffmpeg.org/. Accessed: 2025-12-28.

[15] Financial Crimes Enforcement Network (FinCEN). 2024. FinCEN Alert on Fraud Schemes Involving Deepfake Media Targeting Financial Institutions. FinCEN Alert, FIN-2024-Alert004. https://www.fincen.gov/system/files/shared/FinCEN-Alert-DeepFakes-Alert508FINAL.pdf Defines deepfake media and describes observed fraud typologies and red flags involving synthetic images/audio/video used to bypass verification and enable fraud..

[16] Financial Times. 2024. Arup lost US$25mn in Hong Kong deepfake video conference scam. (2024). https://www.ft.com/content/b977e8d4-664c-4ae4-8a8e-eb93bdf785ea

[17] Google. 2025. Google Meet — Online Video Conferencing. https://workspace.google.com/products/meet/

[18] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. 2018. The eXpress data path: fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (Heraklion, Greece) *(CoNEXT '18)*. Association for Computing Machinery, New York, NY, USA, 54–66. doi:10.1145/3281411.3281443

[19] iperov. [n. d.]. DeepFaceLive: Real-time face swap for PC streaming or video calls. https://github.com/iperov/DeepFaceLive. GitHub repository (archived Nov 13, 2024), accessed 2026-01-02.

[20] Janus. n.d.. Janus: the general purpose WebRTC server. https://github.com/meetecho/janus-gateway. Accessed: 2025-01-14.

[21] Jitsi. [n. d.]. jitsi-videobridge: Jitsi Videobridge (WebRTC SFU). https://github.com/jitsi/jitsi-videobridge. GitHub repository. Accessed: 2025-12-28.

[22] Jitsi. 2026. Jicofo: JItsi COnference FOcus. GitHub repository. https://github.com/jitsi/jicofo Release 2.0.10710 (Jan 14, 2026).

[23] Jitsi. 2026. Jitsi Meet. GitHub repository. https://github.com/jitsi/jitsi-meet

[24] Kurento. n.d.. Kurento Media Server (KMS). https://doc-kurento.readthedocs.io/en/latest/index.html. Accessed: 2025-01-14.

[25] Romeo Lanzino, Federico Fontana, Anxhelo Diko, Marco Raoul Marini, and Luigi Cinque. 2024. Faster Than Lies: Real-time Deepfake Detection using Binary Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 3771–3780.

[26] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. 2020. Face X-Ray for More General Face Forgery Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. 2020. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[28] linux-rdma project. [n. d.]. rdma-core: libibverbs documentation. https://github.com/linux-rdma/rdma-core/blob/master/Documentation/libibverbs.md. Accessed 2026-01-15.

[29] Lynckia. n.d.. Licode: Open source WebRTC communications platform. https://lynckia.com/licode/. Accessed: 2025-01-14.

[30] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. 2025. Scalable Video Conferencing Using SDN Principles. In *Proceedings of the ACM SIGCOMM 2025 Conference* (São Francisco Convent, Coimbra, Portugal) *(SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 1213–1231. doi:10.1145/3718958.3750489

[31] Microsoft. 2020. Remote work trend report: meetings. URL not provided in manuscript (add url/urldate if available).

[32] Microsoft. 2025. Microsoft Teams — Video Conferencing, Meetings, Calling. https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

[33] National Center for Education Statistics (NCES). 2021. Fast Facts: Distance learning. https://nces.ed.gov/fastfacts/display.asp?id=80

[34] National Center for Health Statistics (NCHS), CDC. 2022. Telemedicine Use Among Adults: United States, 2021. https://www.cdc.gov/nchs/products/databriefs/db445.htm

[35] NVIDIA. [n. d.]. NVIDIA Video Codec SDK. https://developer.nvidia.com/video-codec-sdk. Accessed: 2025-12-28.

[36] NVIDIA. [n. d.]. *Scalable Function.* Accessed on January 4, 2025.

[37] NVIDIA Corporation. 2021. *NVIDIA BlueField-3 DPU: Programmable Data Center Infrastructure-on-a-Chip*. Technical Report. NVIDIA Corporation. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf Datasheet (APR21).

[38] NVIDIA Corporation. 2025. *DOCA AES-GCM.* https://docs.nvidia.com/doca/sdk/doca-aes-gcm/index.html DOCA Documentation v3.2.1 LTS. Last updated on Dec. 18, 2025.

[39] NVIDIA Corporation. 2025. *GPUDirect RDMA Documentation.* https://docs.nvidia.com/cuda/gpudirect-rdma/ Last updated Dec. 02, 2025.

[40] NVIDIA Corporation. 2025. NVIDIA Video Codec SDK. https://developer.nvidia.com/video-codec-sdk. Accessed: 2025-01-15.

[41] OpenSSL Project. [n. d.]. OpenSSL: Cryptography and SSL/TLS Toolkit. https://openssl.org/. Accessed: 2026-01-15.

[42] People. 2025. L.A. Woman Loses Life Savings After Scammers Use AI to Pose as *General Hospital* Star, Says Family. (2025). https://people.com/woman-loses-life-savings-after-scammers-use-ai-to-pose-as-general-hospital-star-11800052

[43] Pew Research Center. 2025. How COVID-19 changed U.S. workplaces. https://www.pewresearch.org/politics/2025/02/12/how-covid-19-changed-u-s-workplaces/

[44] George Politis, Boris Grozev, Paweł Domas, Emil Ivov, and Thomas Noël. 2018. Experimental Evaluation of Dynamic Switching between One-on-One and Group Video Calling. In *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*. 1–7. doi:10.1109/IPTCOMM.2018.8567640

[45] Linux Foundation Collaborative Projects. [n. d.]. *Data Plane Development Kit.* Accessed on January 4, 2025.

[46] Linux Foundation Collaborative Projects. 2026. *Open vSwitch.* Accessed on January 4, 2026.

[47] Lee Rainie. 2010. Video calling and video chat. Pew Research Center. https://www.pewresearch.org/internet/2010/10/13/video-calling-and-video-chat/ Reports that 19% of American adults had tried video calling or video chat (online and/or on cell phones)..

[48] Reuters. 2020. Zoom says it has 300 million daily meeting participants, not users. https://www.reuters.com/article/business/zoom-says-it-has-300-million-daily-meeting-participants-not-users-idUSKBN22C1IE/

[49] Reuters. 2025. Italian police freeze cash from AI-voice scam that targeted business leaders. (2025). https://www.reuters.com/technology/artificial-intelligence/italian-police-freeze-cash-ai-voice-scam-that-targeted-business-leaders-2025-02-12/

[50] Kaede Shiohara and Toshihiko Yamasaki. 2022. Detecting Deepfakes with Self-Blended Images . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 18699–18708. doi:10.1109/CVPR52688.2022.01816

[51] Zekun Sun, Yujie Han, Zeyu Hua, Na Ruan, and Weijia Jia. 2021. Improving the Efficiency and Robustness of Deepfakes Detection through Precise Geometric Features. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3608–3617. doi:10.1109/CVPR46437.2021.00361
[52] The Chosun Daily. 2025. Voice Phishing Damage Surpasses 1 Trillion Won in 10 Months. https://www.chosun.com/english/national-en/2025/11/15/D7UMJKGRRZCSRAF52SA6YRLZD4/ Reports that voice-phishing damage in South Korea totaled 1.0566 trillion won from January to October 2025, citing the Korean National Police Agency..
[53] The Korea Herald. 2025. Graphic News: 2025 voice phishing losses surpass W1tr. https://www.koreaherald.com/article/10631333 Cites Korean National Police Agency statistics, including losses of 447.2 billion won (2023) and 854.5 billion won (2024), and reporting that losses surpassed 1 trillion won in 2025..
[54] The Sun. 2026. Shock Moment Filter Glitch Reveals Beauty Influencer's Real Face During Livestream — Causing 140k Fans to Unfollow Her. https://www.the-sun.com/news/15972695/beauty-influencer-real-face-revealed-livestream-unfollows/. Accessed: 2026-06-06.
[55] The WebM Project. 2024. libvpx: VP8/VP9 Codec SDK. https://chromium.googlesource.com/webm/libvpx. Accessed: 2025-01-15.
[56] Tuan Tran, SMH Hosseini, Seyeon Kim, Kyunghan Lee, Nam Bui, Dirk Grunwald, and Sangtae Ha. 2026. {eXpressSFU}: Toward {Super-Scalable} Video Conferencing with {SmartNICs}. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. 2355–2369.
[57] Justin Uberti, Stefan Holmer, Magnus Flodman, Danny Hong, and Jonathan Lennox. 2021. *RTP Payload Format for VP9 Video*. Internet-Draft draft-ietf-payload-vp9-16. IETF. https://datatracker.ietf.org/doc/html/draft-ietf-payload-vp9#section-4.2.1 Work in Progress.
[58] UNESCO. 2023. Startling digital divides in distance learning emerge. URL not provided in manuscript (add url/urldate if available).
[59] W3C. 2024. Scalable Video Coding (SVC) Extension for WebRTC. W3C Recommendation Track document. https://www.w3.org/TR/webrtc-svc/.
[60] Zhendong Wang, Jianmin Bao, Wengang Zhou, Weilun Wang, and Houqiang Li. 2023. AltFreezing for More General Video Face Forgery Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4129–4138.
[61] Magnus Westerlund and Ben Burman. 2015. RTP Topologies. RFC 7667, IETF. https://www.rfc-editor.org/rfc/rfc7667.html
[62] World Health Organization, Regional Office for Europe. 2024. The rise of telehealth in the European Region: insights from Norway. URL not provided in manuscript (add url/urldate if available).
[63] World Wide Web Consortium (W3C). 2025. WebRTC: Real-Time Communication in Browsers. https://www.w3.org/TR/webrtc/
[64] Hanqing Zhao, Tianyi Wei, Wenbo Zhou, Weiming Zhang, Dongdong Chen, and Nenghai Yu. 2021. Multi-attentional Deepfake Detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2185–2194. doi:10.1109/CVPR46437.2021.00222
[65] Yinglin Zheng, Jianmin Bao, Dong Chen, Ming Zeng, and Fang Wen. 2021. Exploring Temporal Coherence for More General Video Face Forgery Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 15044–15054.
[66] Zoom. 2021. A Year Later: Reflecting and Looking Ahead. https://www.zoom.com/en/blog/reflecting-looking-ahead/
[67] Zoom Video Communications. 2025. Zoom Meetings (Virtual Meetings) — Product Page. https://www.zoom.com/en/products/virtual-meetings/
[68] Zoom Video Communications, Inc. [n. d.]. Using end-to-end encryption (E2EE) in Zoom meetings. Zoom Support. https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0065408 Accessed: 2025-12-21.

Appendices

A Statistical Procedure for Onset Detection

The detection process relies on two key indicators: the mean residual energy ($\bar{E}_f$) and cluster score ($\bar{C}_f$). For each incoming frame f with M superblocks, these indicators are calculated as follows:

Localized non-parametric statistical test. When applying statistical methods based on residual energy values, the assumption of normality does not hold because the difference between the values in the normal state and the deepfake synthesis state is large (Figure 8a). To address this, we employ a statistical test to construct an empirical null distribution based on previous frame values. This process involves the following steps to verify the anomaly:

- **Step 1.** Set the mean residual energy of the current frame, $\bar{E}_f$, as the test statistic:
$$\bar{E}_f = \frac{1}{M}\sum_{i=1}^{M} E_{f,i} \tag{1}$$
where i is the superblock index and M is the total number of superblocks.
- **Step 2.** Construct a local null distribution by generating N resampled values based on the mean residual energy values of the previous 30 frames:
$$\mathcal{D}_{local} = \{\bar{E}^*_1, \bar{E}^*_2, \ldots, \bar{E}^*_N\} \tag{2}$$
- **Step 3.** Rearrange the obtained mean values in ascending order to determine the reference distribution:
$$\bar{E}^*_{(1)} \le \bar{E}^*_{(2)} \le \cdots \le \bar{E}^*_{(N)} \tag{3}$$
- **Step 4.** Compare $\bar{E}_f$ with the threshold value $\bar{E}^*_{(k)}$ corresponding to the top 5% ($\alpha = 0.05$), where k is the smallest integer satisfying $k \ge N(1-\alpha)$.
- **Step 5.** If $\bar{E}_f > \bar{E}^*_{(k)}$, the null hypothesis ($H_0$)—that the frame follows natural encoding patterns—is rejected ($p < 0.05$).
- **Step 6.** To isolate temporally distinct anomalies, the candidate frame is validated against a local maximum condition, ensuring it represents the peak value within the temporal neighborhood:
$$\bar{E}_f \ge \max(\bar{E}_{f-1}, \bar{E}_{f-2}, \ldots, \bar{E}_{f-30}) \tag{4}$$

Consequently, the frame is identified as a statistically significant anomaly and selected as a potential deepfake synthesis point.

Adaptive cluster score threshold. After filtering the candidates through the previous step, we utilize the average cluster score, denoted as $\bar{C}_f$, to identify the final deepfake frames.

$$\bar{C}_f = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}(E_{f,i} > E_{threshold}) \tag{5}$$

By incorporating the standard deviation of historical cluster scores, the threshold is adaptively adjusted, allowing the model to dynamically recalibrate its sensitivity. The adaptive threshold T for the current frame f is defined as follows:

$$T_f = T_{base} + w \times \sigma(\bar{C}_{f-1}, \bar{C}_{f-2}, \ldots, \bar{C}_{f-30}) \tag{6}$$

where T_{base} is the video-specific baseline calculated as the mean cluster score of the first 30 frames, w is the weight factor (1.5) based on Tukey's Fences, and $\sigma(\cdot)$ denotes the standard deviation of the cluster scores from the previous 30 frames. A frame is identified as a final deepfake point only if its cluster score $\bar{C}_f$ satisfies the condition:

$$\bar{C}_f \ge T_f \tag{7}$$

B Failure-Mode Analysis

Given the importance of minimizing False Negatives, we carefully analyze the 115 missed detections. Notably, these false negatives are distributed across different videos rather than concentrated within a small subset of particularly difficult cases. This characteristic is especially important in practical scenarios where filter glitches may cause the deepfake effect to disappear and reappear a small number of times during a session [54]. Therefore, missing a single onset transition does not necessarily imply failure to detect the manipulated video.

Table 8: Video-Level Failure Analysis.

Miss0	Miss1	Miss2	Miss≥3	Obs. Fail.	Recall (p)	Theo. Fail. (P_{fail})
886	113	1	0	0% (0/1000)	97.6%	0.000000796%

Since each video contains up to five onset transitions and face-swap pipelines process frames independently, the probability of completely missing a manipulated video can be approximated by the probability of missing all of its onset transitions:

$$P_{\text{fail}} = (1 - p)^5 \tag{8}$$

where p denotes the onset recall. As summarized in Table 8, the measured onset recall is 97.6%, yielding a theoretical video-level failure probability of only 0.000000796%. This theoretical estimate closely matches the empirical observations: among the 1,000 evaluated videos, no video missed all of its onset transitions, resulting in an observed video-level failure rate of 0/1000. Therefore, despite occasional frame-level misses, **no video would be missed entirely**. This observation is further supported by the fact that face-swap pipelines process frames independently, producing detectable inter-frame discontinuities at each onset transition [9].