KVISE: SELECTIVE KV SHARING FOR EFFICIENT DISTRIBUTED LLM COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Communication overhead can become a major bottleneck in collaborative LLM inference when agents exchange large intermediate states. Although KV-cache sharing enables direct reuse of sender-side computation, exhaustively transferring remote caches is often inefficient because KV utility is highly non-uniform and depends on the receiver's current computation. We present **KVise**, a training-free and plug-and-play framework that formulates inter-agent KV exchange as receiver-conditioned communication allocation. The receiver provides compact native attention queries, while senders summarize candidate KV shards using mergeable online-softmax statistics, enabling their utility to be estimated before transmitting raw tensors. KVise then selects shards according to their marginal attention-distortion reduction per byte and jointly reuses the selected states with the receiver's local cache. Across four model families and four multi-hop QA datasets, KVise reduces logical wire traffic by 30.2–61.2% and mean TTFT and completion latency by approximately 21% under our main network setting. Further analyses show concentrated receiver-conditioned KV utility, a controllable quality–communication trade-off, and diminishing latency gains as communication becomes less dominant. These results demonstrate that selective, receiver-aware KV transfer can substantially reduce unnecessary inter-agent communication while preserving flexible control over efficiency and task quality.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed in distributed multi-agent systems, where agents process different parts of a complex task and exchange intermediate results before producing a final response Du et al. (2024); Ye et al. (2025). Communication directly affects both task quality and end-to-end latency, especially when agents operate across edge devices and cloud servers under constrained or heterogeneous network conditions. For example, autonomous systems in underground mines must operate over challenging wireless links, where reliable low-latency communication is critical for remote and collaborative operation Hasan et al. (2025). Consumer-facing agents may similarly involve collaboration between resource-constrained on-device models and more capable cloud models, making communication efficiency an important systems consideration Narayan et al. (2025). Underwater multi-agent systems face even more restrictive communication conditions, including low bandwidth, long propagation delays, and severe channel attenuation Guo et al. (2024). Efficient communication is therefore essential for practical distributed collaborative inference.

Most existing multi-agent systems communicate through natural-language or structured-text messages. Although portable and interpretable, text requires senders to serialize intermediate computation into discrete messages and receivers to process them through additional model execution. This process may discard fine-grained internal information, and message generation itself adds latency Pham et al. (2024); Ramesh & Li (2025). Recent work has therefore explored lower-level communication substrates, including hidden representations and key–value (KV) caches, that enable compatible model instances to reuse intermediate computation more directly.

Direct KV-cache exchange, however, incurs substantial communication and storage overhead (Liu et al., 2024b). Existing work reduces these costs through KV-cache quantization (Li et al., 2025) and through compression, sparsification, and eviction (Behnam et al., 2025). These techniques demon-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Qualitative comparison of representative inter-agent communication approaches.

Method	Training-free	Calibration-free	Receiver-conditioned	Resource-aware	Multi-agent
NLD Du et al. (2024)	✓	✓	×	×	✓
CIPHER Pham et al. (2024)	✓	✓	×	×	✓
AC* Ramesh & Li (2025)	✓	✓	×	×	✓
KVComm Shi et al. (2026)	✓	×	×	△	✓
KVise (Ours)	✓	✓	✓	✓	✓

△ denotes partial support. *For AC, training-free refers to its non-learned activation-combination variants.

strate that substantial KV redundancy can be removed without proportionally degrading model utility.

A related question has received comparatively limited attention in prior research: ***Can we reduce the amount of KV cache transmitted at the source while preserving the task performance of collaborating agents?*** Prior work on attention and KV-cache processing has shown that KV states do not contribute uniformly to downstream computation: useful information is often concentrated in a subset of states, and their importance can vary substantially with the current query. Moreover, lightweight per-block statistics can be used to estimate query-dependent KV importance without accessing every KV entry Tang et al. (2024). Recent work has further explored selective KV sharing between collaborating LLMs Shi et al. (2026). These observations suggest that transmitting the entire sender cache may be unnecessary for effective collaboration and motivate selecting and transferring only the KV states most relevant to the receiver's current task.

We present **KVise**, a *general-purpose, training-free, and plug-and-play* KV-cache communication framework for bandwidth-efficient distributed collaborative inference. KVise formulates inter-agent KV exchange as a receiver-conditioned subset-selection problem without requiring parameter updates or task-specific training. The receiver extracts a compact set of native attention queries. Each sender evaluates its candidate KV shards against these queries and returns compact, mergeable online-softmax statistics instead of full tensor payloads. KVise estimates the attention distortion of candidate subsets and prioritizes shards according to their marginal distortion reduction per transmitted byte under explicit communication and latency constraints.

Only the selected K/V tensors are transferred. KVise aligns the received keys to a consistent positional frame and integrates the transferred tensors with the receiver's local cache through joint attention. This enables direct reuse of task-relevant sender computation. Separating lightweight pre-transfer utility estimation from expensive KV payload transfer avoids transmitting low-value and redundant KV states while preserving the information most relevant to the receiver's current task.

Contributions. Our contributions are:

- **A general-purpose, training-free, plug-and-play framework.** We introduce KVise for bandwidth-efficient communication among compatible LLM agents, formulating inter-agent KV exchange as receiver-conditioned shard selection under explicit communication constraints.
- **Pre-transfer utility estimation and KV reuse.** We use exactly mergeable online-softmax statistics to estimate the marginal utility of candidate KV shards before payload transfer. Selected shards are position-aligned and reused through joint attention with the receiver's local cache.
- **Empirical evaluation.** Across multiple datasets, domains, and model families, KVise maintains competitive task quality relative to full-KV transmission while substantially improving communication efficiency. By selectively transmitting high-value KV states, KVise reduces wire traffic, uses available bandwidth more efficiently, and lowers both time-to-first-token (TTFT) and end-to-end latency.

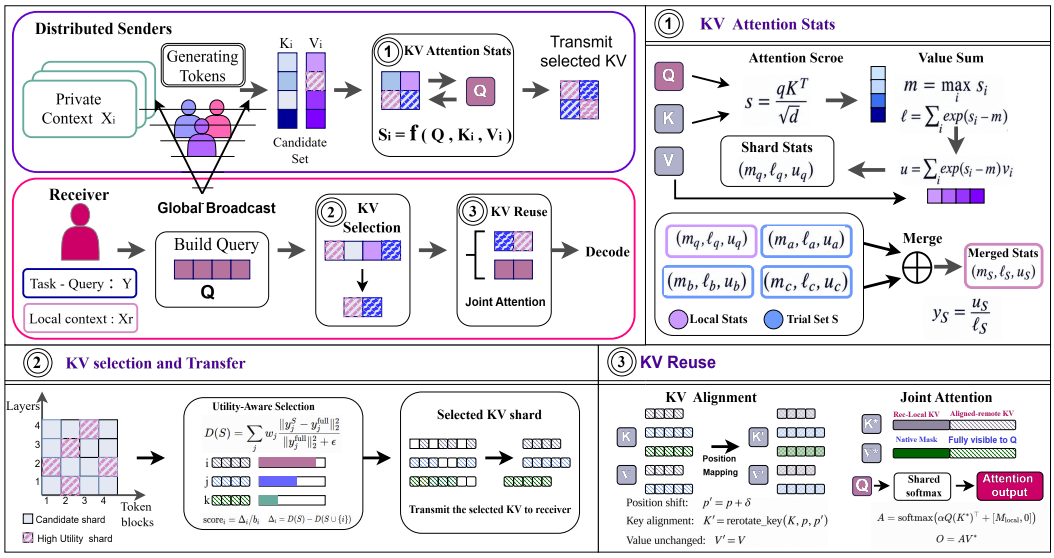


Figure 1: **Overview of KVise. ① KV attention statistics:** the receiver provides compact attention queries, and each sender evaluates its candidate KV shards locally and returns mergeable attention statistics instead of raw KV tensors. **② KV selection and transfer:** KVise estimates receiver-conditioned shard utility, selects the remote states that best preserve the receiver's attention behavior, and transfers only their KV payloads. **③ KV reuse:** the received keys are position-aligned and combined with the receiver's local KV states through joint attention for subsequent inference.

2 METHODOLOGY

We consider collaborative inference among compatible LLM agents that can reuse remote KV states. KVise formulates inter-agent KV exchange as a receiver-conditioned communication-allocation problem: before transmitting KV payloads, it estimates shard utility from compact attention statistics and selects states according to their marginal benefit per transmitted byte. Only the selected K/V tensors are transferred, position-aligned, and jointly reused with the receiver's local cache. KVise is training-free, plug-and-play, and orthogonal to existing KV compression techniques.

2.1 OVERVIEW

KVise coordinates one receiver with one or more compatible sender agents. Each sender independently processes its private context and partitions the resulting KV states into a candidate shard set $\mathcal{C}$, which serves as the unit of remote KV selection and transfer. In parallel, the receiver performs a provisional forward pass over its local context and current task query to extract a compact query bank $\mathcal{Q} = \{(q_p, w_p)\}_{p=1}^{P}$, where q_p is a native attention query and w_p specifies its contribution to the selection objective. These queries condition remote-KV evaluation on the receiver's current computation, rather than assigning each sender-side shard a fixed importance score.

As shown in Figure 1, KVise separates *KV evaluation* from *KV payload transfer*. The receiver first broadcasts its query probes to the participating senders. Each sender evaluates its local shards against the probes and returns compact, mergeable attention statistics instead of immediately transmitting candidate K/V tensors. The receiver composes these statistics to estimate the attention response induced by different candidate subsets without fetching their raw K/V payloads. It selects a subset $S^\star \subseteq \mathcal{C}$ that preserves the relevant full-KV attention behavior with reduced communication.

Only the K/V tensors in $S^\star$ are subsequently transferred. Their positional representations are mapped into the receiver's inference frame, and the tensors are combined with the receiver-local cache through joint attention. The provisional receiver state used for selection is then discarded, and the corresponding suffix is replayed with the selected remote KV states to produce the formal inference result. KVise follows a one-shot pipeline of *receiver probing, pre-transfer KV selection, selective transfer, and position-aligned joint reuse*.

2.2 KV SELECTION

KVise selects remote shards by preserving the receiver's current attention response with minimum KV communication. Let b_c denote the payload size of candidate shard c and $B(S) = \sum_{c \in S} b_c$ the payload of a subset $S \subseteq \mathcal{C}$. Given a distortion tolerance τ, the selection problem is

$$S^\star = \arg\min_{S \subseteq \mathcal{C}} B(S) \qquad \text{s.t.} \qquad D(S) \le \tau, \tag{1}$$

where $D(S)$ measures the deviation between the attention response induced by S and that induced by the complete candidate set. The challenge is to evaluate $D(S)$ before the corresponding K/V tensors are transferred. For a receiver probe q and a candidate shard (K, V), the sender first computes the scaled attention logits $z = qK^\top/\sqrt{d_h}$ and summarizes the resulting attention computation using the numerically stable online-softmax state

$$\phi = (m, \ell, u), \qquad m = \max_i z_i, \quad \ell = \sum_i e^{z_i - m}, \quad u = \sum_i e^{z_i - m} v_i.$$

Probe and shard indices are omitted for clarity. The compact state retains the quantities required to recover the corresponding attention output as $\mathcal{R}(\phi) = u/\ell$. States computed over disjoint KV shards can be merged exactly after the appropriate softmax renormalization. Thus, statistics from multiple remote shards can be composed with the receiver-local state without exposing the remote tensor payloads to the receiver. The exact merge rule and its derivation are deferred to the appendix. Let ϕ_p^L denote the receiver-local state associated with probe p and $\oplus$ the merge operator. KVise evaluates a candidate subset using

$$\begin{aligned} A_p(S) &= \mathcal{R}\left(\phi_p^L \oplus \bigoplus_{c \in S} \phi_{p,c}\right), \\ D(S) &= \sum_{p=1}^{P} w_p \frac{\|A_p(S) - A_p(\mathcal{C})\|_2^2}{\|A_p(\mathcal{C})\|_2^2 + \epsilon}. \end{aligned} \tag{2}$$

Here, $A_p(\mathcal{C})$ is the full-candidate reference for probe p. Thus, $D(S)$ measures how well a selected subset reproduces the attention behavior obtained when all candidate KV states are available. A shard's contribution can vary with the receiver probes and the already-selected shards, so KVise treats shard utility as receiver- and subset-dependent. For small candidate sets, KVise directly evaluates feasible subsets and returns the minimum-payload solution satisfying Equation 1. For larger candidate spaces, it constructs the subset iteratively using the conditional distortion reduction per transmitted byte,

$$\rho(c \mid S) = \frac{D(S) - D(S \cup \{c\})}{b_c}.$$

Shards with larger $\rho(c \mid S)$ provide greater attention preservation for the same communication cost and are therefore prioritized until the distortion tolerance is reached. All subset evaluation uses the compact attention statistics; the selected raw K/V tensors are requested only after $S^\star$ is determined. The distortion criterion operates in the provisional probe space and is a pre-transfer surrogate for downstream KV utility.

2.3 KV REUSE

After determining $S^\star$, KVise fetches only the corresponding remote K/V tensors. Remote keys are generated under sender-side token positions, whereas reuse occurs in the receiver's replay frame. To address this positional compatibility issue, KVise re-expresses each selected key under its assigned receiver position before inserting it into the receiver cache. For the RoPE-based compatible models considered in our implementation, let k_p denote a key encoded at sender position p, and let p' denote its mapped position in the receiver frame. Positional alignment is expressed as

$$\widetilde{k}_{p'} = R(p')R(p)^{-1}k_p, \qquad \widetilde{v} = v,$$

where $R(\cdot)$ denotes the rotary transformation. Thus, only the positional representation of the key is changed; its associated value is preserved. The full position-mapping procedure is provided in the

appendix. At layer ℓ, the aligned remote states are concatenated with the receiver-local cache,

$$K_\ell^\star = \left[K_\ell^L; \widetilde{K}_{\ell,S^\star}^R\right], \qquad V_\ell^\star = \left[V_\ell^L; V_{\ell,S^\star}^R\right],$$
$$H_\ell^\star = \mathrm{softmax}\left(\frac{Q_\ell (K_\ell^\star)^\top}{\sqrt{d_h}} + M_\ell\right) V_\ell^\star. \tag{3}$$

This is a joint attention operation rather than a post-hoc fusion of two independent attention outputs: local and remote keys compete under the same softmax normalization, allowing the receiver's native queries to determine how much information to draw from each source. The provisional suffix used to construct the query bank is not retained as formal inference state. Before reuse, the receiver restores its stable local cache, mounts the aligned remote shards at their corresponding layers, and replays the suffix under the augmented KV context. The resulting states then enter the normal autoregressive decoding process. This replay avoids committing computation performed before remote-KV selection while preserving the receiver model's native attention and decoding procedure.

Summary. KVise separates the decision of *what to communicate* from the transfer of the KV payload. Receiver-native queries and mergeable attention statistics enable pre-transfer evaluation of remote shards, after which only a utility-preserving subset is transmitted, position-aligned, and incorporated through joint attention. The procedure is training-free and requires no modification to the underlying model parameters.

3 Experiments

We evaluate whether KVise improves bandwidth utilization in distributed LLM-agent collaboration by prioritizing remote KV states useful to the receiver. We assess communication reduction and task quality, report calibrated inference-latency estimates, and examine the bandwidth–quality trade-off and remote KV-shard utility.

Experimental setup. Experiments were conducted on a server equipped with four NVIDIA RTX A6000 GPUs (48 GB each), two AMD EPYC 7H12 64-Core processors, and 503 GiB of system memory, running Ubuntu 20.04.5. The software stack comprised Python 3.11.15, PyTorch 2.6.0, and Transformers 5.15.1. All models were evaluated in bfloat16 precision using the native PyTorch/Transformers scaled dot-product attention (SDPA) backend.

We evaluate KVise on four multi-hop question-answering datasets: 2WikiMQA (Ho et al., 2020), MedHop (Welbl et al., 2018), IIRC (Ferguson et al., 2020), and HotpotQA-distractor (Yang et al., 2018). To evaluate generality across model families and parameter scales, we use Qwen3-8B (Yang et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Ministral-3-14B-Instruct-2512 (Liu et al., 2026), and Phi-4 (14B) (Abdin et al., 2024). The main evaluation covers paired samples across 16 model–dataset combinations, with each sample evaluated under both KVise and Exhaustive Transfer. Unless otherwise specified, all experiments use a two-sender, one-receiver (2S1R) topology with a network round-trip time (RTT) of 1 ms. Task quality is measured using each dataset's native evaluation metric.

3.1 Overall Performance

We compare KVise with Exhaustive Transfer, which transmits the complete sender-generated KV cache without selective filtering. We evaluate bandwidth utilization, task quality, and inference latency. In this main evaluation, the three agents share an aggregate 10 Mbps full-duplex network link. KVise uses minimum-byte selection with a distortion threshold of $\tau = 0.02$ and partitions sender-generated KV states into 32-token shards. Task quality is reported using each dataset's native metric: normalized token F1 for 2WikiMQA, candidate accuracy for MedHop, DROP F1 for IIRC-lite, and answer F1 for HotpotQA-distractor-lite. Communication efficiency is measured by logical wire traffic. Latency is estimated through calibrated trace replay and reported using time to first token (TTFT) and completion latency (CL). Wire, TTFT, and CL are reported as relative reductions over Exhaustive Transfer on paired samples with matched sender generations. Latency reductions compare means within each model–dataset setting.

Table 2: KVise versus Exhaustive Transfer: Efficiency and Quality.

Dataset	Model	Quality Exhaustive → KVise	Wire Red. ↑	TTFT Red. ↑	CL Red. ↑
2WikiMQA Token F1	Qwen3-8B	0.4953 → 0.4439	43.71%	12.51%	12.46%
	Llama-3.1-8B	0.2101 → 0.1902	41.76%	18.62%	18.41%
	Ministral-3-14B	0.4519 → 0.4227	30.19%	8.11%	8.13%
	Phi-4-14B	0.2442 → 0.1567	61.16%	34.82%	33.14%
MedHop Accuracy	Qwen3-8B	0.1462 → 0.1287	54.69%	27.64%	26.66%
	Llama-3.1-8B	0.1023 → 0.1023	46.10%	25.41%	25.47%
	Ministral-3-14B	0.1023 → 0.1082	46.12%	14.80%	14.76%
	Phi-4-14B	0.0936 → 0.1140	52.93%	29.96%	29.95%
IIRC DROP F1	Qwen3-8B	0.2366 → 0.2875	54.04%	21.62%	21.98%
	Llama-3.1-8B	0.3181 → 0.2891	38.43%	17.18%	17.13%
	Ministral-3-14B	0.2443 → 0.2849	36.02%	11.11%	11.12%
	Phi-4-14B	0.2529 → 0.2812	54.53%	28.72%	28.41%
HotpotQA Answer F1	Qwen3-8B	0.5681 → 0.4879	49.30%	24.68%	24.53%
	Llama-3.1-8B	0.4048 → 0.4002	39.34%	20.59%	20.44%
	Ministral-3-14B	0.5206 → 0.4578	35.02%	13.37%	13.40%
	Phi-4-14B	0.3915 → 0.3174	54.47%	32.17%	31.68%

Table 2 reports results under weak-network conditions, assessing communication and latency savings alongside task-quality loss. Across 16 model–dataset settings, KVise reduces logical wire traffic by 30.19–61.16% (46.11% on average), TTFT by 8.11–34.82% (21.33% on average), and end-to-end completion latency by 8.13–33.14% (21.10% on average), with both latency reductions based on calibrated estimates. Selective KV transmission therefore reduces communication volume and estimated latency on the inference critical path in these settings. The similar TTFT and completion-latency reductions are consistent with benefits along the communication-sensitive prefill and first-token path that persist through subsequent generation.

These gains involve a model–task-dependent quality–communication trade-off. On MedHop and IIRC, several settings have quality point estimates equal to or above those of Exhaustive Transfer while reducing logical wire traffic by 36–55%, whereas some 2WikiMQA and HotpotQA settings exhibit larger quality decreases. This variation suggests that transferred KV volume alone is an unreliable proxy for utility and that receiver–task pairs tolerate different levels of KV reduction. KVise therefore acts as a bandwidth-allocation rather than cache-compression mechanism, prioritizing receiver-relevant KV states when allocating communication resources.

Figures 2–4 provide further diagnostics. Wire reduction is strongly correlated with estimated completion-latency reduction ($r = 0.89$) under weak-network conditions. Quality remains model–task dependent: 11 of 16 settings have 95% confidence intervals for quality differences overlapping zero, with clearer degradations concentrated in subsets of 2WikiMQA and HotpotQA, motivating receiver-aware rather than uniform KV reduction. Control overhead, expressed as KVise control-plane bytes relative to paired Exhaustive Transfer logical wire traffic, remains around the percent level and below approximately 2.6% for most observed requests, with a small number exceeding this level. The typical overhead is small relative to the 30–61% logical wire reduction. These observations suggest that control overhead leaves most communication savings from selection intact, alongside lower estimated TTFT and end-to-end latency and a task-dependent quality trade-off.

3.2 Distortion-Controlled Trade-off

We evaluate the communication–quality trade-off by sweeping the distortion threshold τ on Llama-3.1-8B-Instruct with all 300 HotpotQA-distractor samples. As shown in Table 3, increasing τ raises logical-wire reduction from 29.70% to 73.26%, but increasingly aggressive selection eventually degrades answer quality. In particular, $\tau = 0.02$ reduces wire traffic by 42.70% while retaining an Answer F1 point estimate close to Exhaustive Transfer, whereas larger thresholds yield additional

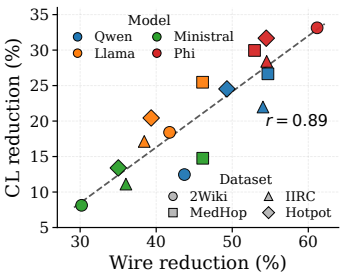


Figure 2: Wire reduction versus CL reduction.

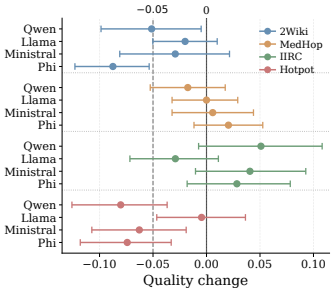


Figure 3: Quality change with 95% CIs.

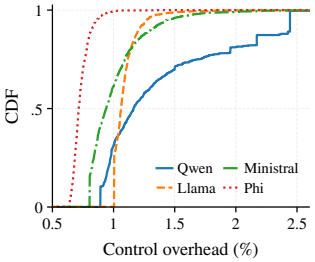


Figure 4: Control overhead distribution.

Table 3: Quality–communication trade-off under different distortion thresholds.

Setting	**Answer F1** ↑	**Answer EM** ↑	**Wire Red.** ↑
Exhaustive	0.4084	27.00%	–
$\tau = 0.01$	0.4111 (+0.0027)	26.67% (−0.33)	29.70%
$\tau = 0.02$	0.4118 (+0.0033)	26.33% (−0.67)	42.70%
$\tau = 0.04$	0.3882 (−0.0202)	25.00% (−2.00)	57.58%
$\tau = 0.08$	0.3235 (−0.0850)	20.67% (−6.33)	73.26%

communication savings at progressively greater quality cost. These results identify τ as a direct control knob for navigating the quality–communication trade-off, with $\tau = 0.02$ providing a practical balance in this sweep.

3.3 Sensitivity to Network Capacity

We evaluate how KVise responds to changing network capacity using Qwen3-8B on HotpotQA, varying the aggregate bandwidth from 3 to 20 Mbps while keeping the KV selection policy and inference settings fixed. Table 4 reports the resulting TTFT and completion-latency reductions. KVise provides the largest gains at lower bandwidths: at 3 Mbps, it reduces p50 TTFT by 45.89% and mean completion latency by 41.45%. The benefit decreases smoothly as bandwidth increases, reaching 11.43% and 3.94%, respectively, at 20 Mbps. Tail latency follows the same trend and eventually becomes slightly worse at 20 Mbps, indicating that communication optimization has diminishing impact once network delay is no longer dominant.

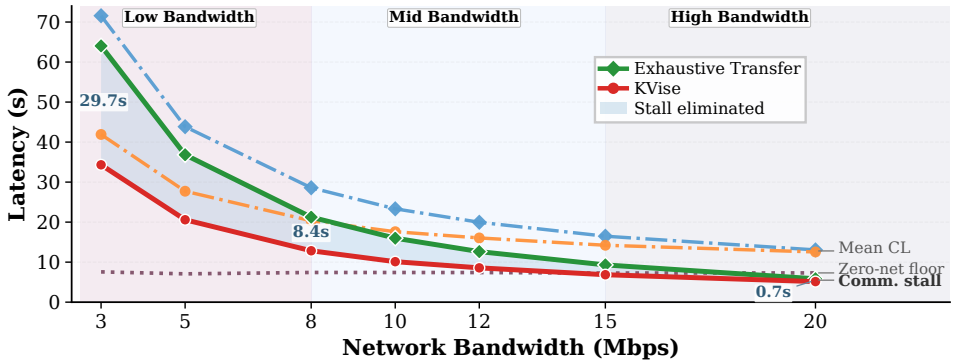


Figure 5: Latency decomposition across network capacities. The shaded region between Exhaustive Transfer and KVise denotes the critical-path communication stall eliminated by selective KV transfer. As bandwidth increases, communication stall contracts toward the zero-network latency floor, leaving less latency headroom for communication optimization.

Table 4: Latency reduction of KVise across network bandwidths on Qwen3-8B and HotpotQA.

Metric	3 Mbps	5 Mbps	8 Mbps	10 Mbps	12 Mbps	15 Mbps	20 Mbps
TTFT p50	45.89%	41.56%	33.34%	31.16%	25.45%	18.82%	11.43%
TTFT p95	36.80%	31.86%	22.14%	15.16%	10.61%	4.32%	−9.16%
CL Mean	41.45%	36.78%	28.83%	24.53%	19.62%	13.72%	3.94%
CL p95	36.73%	31.78%	23.31%	19.28%	10.96%	4.95%	−8.37%

Figure 5 explains this trend. At 3 Mbps, KVise reduces critical-path communication stall from 64.03 s to 34.32 s, removing roughly 29.7 s from the critical path. At 20 Mbps, the corresponding gap contracts to only 0.7 s. Meanwhile, the zero-network latency floor remains nearly unchanged. This shows that KVise's bandwidth-dependent speedup is primarily obtained by removing communication from the request critical path: when communication is expensive, selective KV transfer translates directly into substantial latency savings; as network capacity increases, the remaining compute and protocol costs dominate and bound the attainable gain.

3.4 KV-Shard Utility Concentration

We next examine the utility distribution of remote KV shards as a diagnostic of selective communication. We analyze Qwen3-8B in the IIRC shared-cache setting, with 96 candidate KV shards per task.

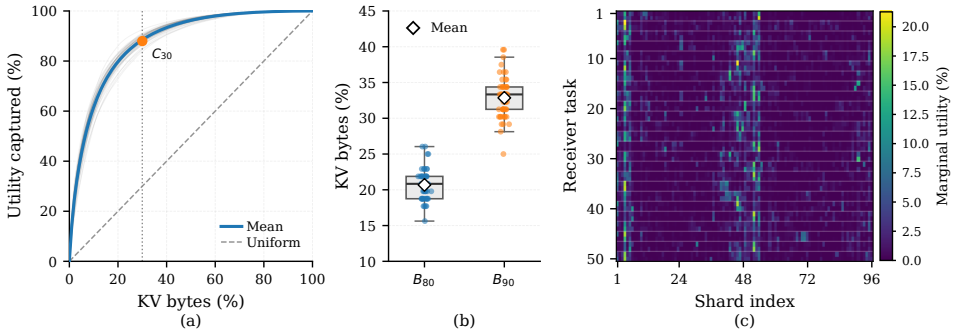


Figure 6: Concentration of receiver-conditioned KV-shard utility. (a) Cumulative utility captured as the transferred KV volume increases. (b) Per-task KV fractions required to capture 80% and 90% of utility. (c) Marginal utility of individual KV shards across receiver tasks.

Figure 6(a) reveals concentrated KV-shard utility in this setting. On average, no more than 30% of the KV bytes capture 87.98% of the measured utility, above the uniform-allocation reference.

Figure 6(b) illustrates the KV budget required to recover a given fraction of utility: 20.69% and 32.85% of the KV bytes are required on average to recover 80% and 90% of the utility, respectively. The corresponding utility distribution has a Gini coefficient of 0.737, indicating substantial non-uniformity across shards.

Figure 6(c) visualizes that high marginal utility is concentrated in a relatively small subset of shard–task combinations, while many shards contribute little to a given receiver task. The locations and magnitudes of these high-utility regions vary across receiver tasks, motivating receiver-conditioned selection rather than exhaustive transmission. In this diagnostic setting, the results support KVise's allocation of communication to a compact subset of high-value KV states instead of uniformly across the complete remote cache.

4 Related Work

Communication efficiency is increasingly important in LLM-based multi-agent systems. Most systems exchange natural-language messages, while recent work reduces communication through mes-

sage pruning or adaptive interaction topologies (Zhang et al., 2025a;b). Other studies move beyond text and communicate model-internal representations, including continuous latent states (Du et al., 2026) and KV caches (Shi et al., 2026). These approaches show that internal representations can provide a more direct communication substrate than generated messages. KVise builds on this direction but targets a different decision: rather than determining *when* agents communicate or *which representation type* they exchange, it determines *which sender-side KV states* should be transmitted for the receiver's current computation, without task-specific training.

KV-cache efficiency has been extensively studied because KV states exhibit substantial redundancy and non-uniform utility. SnapKV and Quest exploit attention patterns to identify important KV states (Li et al., 2024; Tang et al., 2024), while PyramidKV allocates cache budgets unevenly across layers (Cai et al., 2025). KIVI reduces KV storage through quantization (Liu et al., 2024c), MiniCache exploits cross-layer redundancy (Liu et al., 2024a), and InfiniGen predicts useful KV entries to reduce memory movement (Lee et al., 2024). These methods primarily optimize KV storage or access after the relevant cache is locally available. KVise instead applies receiver-conditioned utility estimation *before communication*, using compact attention statistics to allocate transmission to high-value remote KV shards. It is therefore complementary to KV compression and quantization methods, which can further reduce the payload of the selected states.

5 Discussion

Our results suggest that efficient inter-agent KV communication is fundamentally an allocation problem rather than simply a compression problem. KVise exploits the fact that remote KV states contribute unevenly to a receiver's current computation and that this utility can be estimated before transmitting the underlying tensors. Across the evaluated model–dataset settings, selectively allocating communication to high-value shards substantially reduces transferred KV volume and logical wire traffic, with corresponding reductions in TTFT and completion latency. The utility analysis further shows that a relatively small fraction of KV bytes can capture most of the measured receiver-conditioned utility, supporting the central premise that exhaustive KV transfer is often unnecessary.

The experiments also expose the limits of aggressive selection. Communication savings and task quality form a controllable but model- and task-dependent trade-off: increasing the distortion tolerance transmits fewer KV states, but eventually removes information required for downstream reasoning. Similarly, latency improvements depend on how strongly communication contributes to the inference critical path. KVise provides the largest speedups when network transfer is expensive, while its advantage naturally decreases as network capacity increases and computation becomes dominant. These observations indicate that there is no universally optimal amount of KV transfer; instead, the appropriate operating point should reflect both the receiver's task and the deployment environment. Importantly, KVise introduces this control without modifying model parameters and remains compatible with existing KV quantization and compression techniques.

More broadly, KVise demonstrates that intermediate model states need not be treated as monolithic communication payloads. By making their transfer conditional on receiver-side utility, KVise turns KV exchange into an explicit communication-resource allocation problem. This provides a general mechanism for reducing unnecessary inter-agent data movement while retaining direct reuse of internal computation, and offers a practical foundation for more communication-aware collaborative LLM systems.

6 Conclusion

We presented KVise, a training-free framework for communication-efficient KV sharing among compatible LLM agents. KVise treats inter-agent KV exchange as a receiver-conditioned allocation problem, estimating shard utility before payload transfer and transmitting only the states most relevant to the receiver's current computation. Across multiple models and datasets, KVise substantially reduces KV communication and communication-sensitive inference latency, while exposing a controllable, model- and task-dependent quality–communication trade-off. These results show that exhaustive KV transfer is often unnecessary and that selectively allocating communication to high-value intermediate states provides a practical direction for more efficient collaborative LLM inference.

AI USE STATEMENT

In this work, generative AI models were used primarily as experimental subjects in evaluating the proposed method. Generative AI tools were also used to assist with collecting, organizing, and summarizing relevant research materials during the preparation of this work. We did not rely on generative AI to determine experimental outcomes or to replace the authors' analysis of the reported results.

All AI-assisted information collection was reviewed and verified by the authors against the corresponding original sources before being incorporated into the manuscript. All experimental results reported in this work were obtained from our implemented experimental pipeline. The authors take responsibility for the final content of this work, including all claims, results, and artifacts.

ETHICS STATEMENT

This work focuses on communication-efficient KV-cache sharing among compatible large language models and does not involve human subjects, collection of personally identifiable information, or deployment in safety-critical environments. Our experiments use existing language models and established benchmark datasets for research evaluation. We do not introduce new datasets containing personal or sensitive information. The proposed method is intended to improve the communication and inference efficiency of collaborative LLM systems and does not alter the underlying models' safety mechanisms or intended usage constraints. We follow the licenses and terms associated with the models, datasets, and software used in our experiments.

REPRODUCIBILITY STATEMENT

This work is designed to be fully reproducible. We provide complete descriptions of the KVise algorithm, including KV-shard utility estimation, budgeted selection, positional alignment, and receiver-side joint-attention integration, together with the experimental settings, evaluated models and datasets, system topology, communication configurations, baselines, metrics, and fixed experimental parameters used in our evaluation. The complete source code, implementation scripts, experiment configurations, and instructions required to reproduce the reported results will be included in the supplementary material. Additional pseudocode, implementation details, and extended experimental results are also provided in the appendix and supplementary material. Together, these materials enable reproduction of the algorithmic behavior as well as the reported task-quality, communication, and latency results.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report. Technical Report MSR-TR-2024-57, Microsoft Research, 2024. URL https://arxiv.org/abs/2412.08905.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. RocketKV: Accelerating long-context LLM inference via two-stage KV cache compression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3358–3392. PMLR, 2025. URL https://proceedings.mlr.press/v267/behnam25a.html.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. In *Conference on Language Modeling*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine*

Learning Research, pp. 11733–11763. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. IIRC: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1137–1147. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.86. URL https://aclanthology.org/2020.emnlp-main.86/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Jiani Guo, Shanshan Song, Jun Liu, Lei Wan, Yang Yu, and Guangjie Han. An MC-CDMA-based MAC protocol for efficient concurrent communication in mobile underwater acoustic networks. *IEEE Transactions on Mobile Computing*, 23(12):12428–12443, 2024. doi: 10.1109/TMC.2024.3409562.

Shahriar Hasan, Katrin Sjöberg, and Peter Wallin. Wireless communication in underground mining teleoperation: A systematic review. *IEEE Access*, 13:72577–72602, 2025. doi: 10.1109/ACCESS.2025.3563138.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. Infinigen: Efficient generative inference of large language models with dynamic kv cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation*, 2024.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36451–36485. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25dd.html.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems*, 2024a.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026. URL https://arxiv.org/abs/2601.08584.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. CacheGen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*. Association for Computing Machinery, 2024b. doi: 10.1145/3651890.3672274. URL https://doi.org/10.1145/3651890.3672274.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, 2024c.

Avanika Narayan, Dan Biderman, Sabri Eyuboglu, Avner May, Scott Linderman, James Zou, and Christopher Re. Cost-efficient collaboration between on-device and cloud language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45682–45719. PMLR, 2025. URL https://proceedings.mlr.press/v267/narayan25a.html.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A. Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e444859b2a22df6b56af9381ad1e9480-Abstract-Conference.html.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 51094–51116. PMLR, 2025. URL https://proceedings.mlr.press/v267/ramesh25a.html.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire Jr., and Dejan Kostic. KVComm: Enabling efficient LLM communication through selective KV sharing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=F7rUng23nw.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018. doi: 10.1162/tacl_a_00021. URL https://aclanthology.org/Q18-1021/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0605. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1a074a28c3a6f2056562d00649ae6416-Abstract-Conference.html.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, 2025a.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025b.

A Additional Method Details

We provide additional definitions and execution details for the KVise pipeline described in Section 2. Specifically, we describe the construction of candidate KV shards and receiver query probes, distinguish provisional from formal receiver inference, and clarify the scope of the selection objective. These details require no additional training and do not modify the objectives in Equations 1 and 2.

A.1 Collaboration Setting and KV Compatibility

A collaboration episode consists of one receiver R and a set of senders $\mathcal{S} = \{S_1, \ldots, S_M\}$. Each sender processes a private context and generates an intermediate sequence from which reusable KV states are extracted. The receiver maintains its own local context and a task-dependent suffix containing the current query and the information required to produce the final response.

KVise assumes that the participating agents are *KV-compatible*: after positional alignment, the receiver must be able to interpret a remote KV state. Within each evaluation run, all sender and receiver roles use independent instances of the same model checkpoint, with matching attention geometry, tokenizer, numerical precision, and positional-encoding configuration. These roles do not share model-state storage during inference. Although the evaluated model family varies across experimental settings, KV states are exchanged only between compatible instances within a setting.

KVise transfers the original K/V tensors produced during sender inference. Shard selection neither quantizes nor reconstructs their contents. Instead, the approximation lies in deciding *which* remote states to communicate. The selection mechanism is therefore orthogonal to KV quantization and compression methods that could be applied to the selected payloads.

A.2 Candidate KV Shards

For sender S_s, let

$$\mathcal{C}_s = \{c_{s,1}, \ldots, c_{s,N_s}\}$$

denote its committed candidate shards. The complete remote candidate universe is

$$\mathcal{C} = \bigcup_{s=1}^{M} \mathcal{C}_s.$$

This is the candidate set used in Section 2.2. Our implementation applies no additional semantic key index or receiver-independent pre-filter before KVise selection.

Each sender first performs standard autoregressive generation over its private input. Only KV states corresponding to sender-generated tokens are eligible for remote sharing. KV states associated with the sender's private prompt remain local and are excluded from both $\mathcal{C}$ and the communication accounting of the Exhaustive Transfer baseline. KVise and Exhaustive Transfer therefore operate on the same sender-generated KV universe.

The generated KV sequence is divided into contiguous token blocks. A completed block is committed once its K/V tensors and token positions become immutable. The final, potentially shorter block is committed when sender generation terminates. Each committed token block may also be partitioned into contiguous layer groups. A shard c is therefore identified by its sender, generation epoch, layer interval, and token interval, and has an associated payload size b_c. Shards in the same sender-side partition do not overlap in token range.

Only committed shards are eligible for evaluation or transfer. After all senders complete generation, their manifests define an immutable snapshot of the candidate universe $\mathcal{C}$. Selection operates on this snapshot, so that the full-candidate reference $A_p(\mathcal{C})$, candidate payload sizes, and transferred tensors all refer to the same sender states. In the main experiments, KV states are partitioned using blocks of 32 generated tokens.

A.3 Receiver Query-Probe Construction

KVise conditions remote-KV selection on the receiver's current computation through the query bank

$$\mathcal{Q} = \{(q_p, w_p)\}_{p=1}^{P}, \qquad w_p \geq 0, \qquad \sum_{p=1}^{P} w_p = 1,$$

introduced in Section 2.1. The probes are native attention queries produced by the receiver, rather than outputs of a separately trained scoring model.

We divide the receiver input into a stable local prefix and a task-dependent suffix of T tokens. The receiver first prefills the stable prefix and preserves the resulting local cache. It then performs a *provisional* forward pass over the suffix without mounting remote KV states. During this pass, KVise captures native post-RoPE query vectors from a small set of layers, attention heads, and suffix positions.

The default probe policy samples the receiver computation along three axes: up to three layers distributed across model depth, up to four KV-head groups, and up to three positions distributed across the provisional suffix. For grouped-query attention (GQA), a sampled query head is associated with its KV head according to

$$h_{\mathrm{kv}} = \left\lfloor \frac{h_q}{H_q / H_{\mathrm{kv}}} \right\rfloor,$$

where H_q and H_{kv} are the numbers of query and KV heads, respectively. This mapping preserves the model's native query–KV-head association.

Sender and receiver states may be produced at different physical token positions. KVise therefore re-expresses each captured post-RoPE query in a common attention frame before exporting it. This operation changes only the query's positional representation and preserves its native content. The same positional convention is used to evaluate sender-side keys and to mount selected keys at the receiver. Appendix D gives the full derivation.

KVise compacts probes within each layer–query-head stream to avoid repeatedly evaluating similar receiver states. Let η_{dedup} denote the cosine-similarity threshold. Queries whose similarity to a cluster representative exceeds this threshold are grouped, and the medoid query is retained as the representative probe. The weights assigned to queries in the same cluster are accumulated before final normalization. Our implementation uses $\eta_{\mathrm{dedup}} = 0.995$ by default.

Before compaction, a probe extracted from suffix position $t \in \{0, \ldots, T-1\}$ receives the unnormalized weight

$$\widetilde{w}_p = \alpha_{\ell(p)} \frac{t+1}{T},$$

where $\alpha_{\ell(p)}$ is the weight assigned to its probe layer. Unless otherwise specified, all layers receive equal weight, $\alpha_\ell = 1$. The linear position factor assigns greater weight to later suffix states because they incorporate a longer prefix of the receiver's current task computation. After compaction, the cluster weights are summed and the retained weights are normalized:

$$w_p = \frac{\widetilde{w}_p}{\sum_{j=1}^{P} \widetilde{w}_j}.$$

The resulting query bank is compact while retaining the contributions of all sampled receiver states represented by each probe.

A.4 Pre-Transfer Evaluation Boundary

KVise strictly separates candidate evaluation from KV payload transfer. Once $\mathcal{Q}$ and the candidate snapshot $\mathcal{C}$ have been established, the receiver sends the compact probes and candidate identifiers to the senders. Each sender evaluates its local candidate shards against the corresponding receiver probes. It returns only the online-softmax states required by Equation 2, together with the associated candidate metadata. No raw candidate K/V tensors are transmitted at this stage.

For each probe p, the receiver combines its local attention state with the returned per-shard states. These merged states define both the full-candidate reference $A_p(\mathcal{C})$ and the counterfactual response

$A_p(S)$ for any candidate subset $S \subseteq \mathcal{C}$. The exact merge operator is derived in Appendix B. The receiver can therefore compare candidate subsets before requesting their tensor payloads.

The receiver requests raw K/V tensors only after selecting $S^\star$. Consequently, scoring an unselected candidate requires only compact control-plane information and attention statistics, rather than the candidate's full KV payload.

A.5 Provisional and Formal Receiver States

The provisional suffix pass is used only to obtain the receiver-conditioned selection signal. Its hidden states and suffix KV cache do not become part of the formal inference state. This distinction prevents KVise from committing receiver computation performed before the selected remote information became available.

After receiving the K/V tensors in $S^\star$, the receiver maps the remote keys into its inference frame, as described in Section 2.3 and Appendix D. It then restores the stable local cache saved before probing, mounts the selected remote shards, and replays the same receiver suffix. During formal replay, the local and remote K/V states participate in the joint attention operation in Equation 3. The replay states, rather than the provisional states, initialize subsequent autoregressive decoding.

KVise therefore follows this one-shot execution sequence:

sender generation $\rightarrow$ receiver probing $\rightarrow$ attention-state evaluation
$\rightarrow$ KV selection $\rightarrow$ selective KV transfer
$\rightarrow$ formal replay and decoding

Selection finishes before any selected payload is mounted. The standard pipeline does not repeat selection after formal replay.

A.6 Scope of the Selection Surrogate

The distortion $D(S)$ in Equation 2 is a pre-transfer surrogate. It measures how closely a selected remote subset reproduces the full-candidate attention response for the provisional query bank $\mathcal{Q}$. The constraint

$$D(S^\star) \leq \tau$$

therefore specifies fidelity within the sampled receiver probe space.

The constraint does not imply equality at every layer, attention head, token position, future decoding query, or final answer. Receiver queries produced after mounting remote KV states may differ from those in the provisional pass because the remote states affect subsequent receiver computation. KVise thus uses the provisional query bank as a receiver-conditioned selection signal, not as a formal guarantee of downstream task equivalence.

For the same reason, τ controls the communication–quality trade-off rather than guaranteeing task quality. Smaller values require the selected subset to reproduce the full-candidate response more closely in the probe space. Larger values permit more aggressive KV reduction. The relationship between this surrogate, transferred KV volume, and downstream task quality is evaluated empirically in Section 3.2.

B Exact Composition of Online-Softmax Attention States

We next derive the exact merge rule for the online-softmax states used in Section 2.2. Attention states computed independently over disjoint KV shards can be composed without accessing their underlying K/V tensors. For a fixed receiver probe, the merged state recovers the attention response obtained by evaluating the union of those KV states directly.

B.1 Single-Shard Attention State

Consider a fixed receiver probe q and a KV shard containing keys $K = \{k_i\}_{i\in\mathcal{I}}$ and values $V = \{v_i\}_{i\in\mathcal{I}}$. As in Section 2.2, the scaled attention logits are

$$z_i = \frac{qk_i^\top}{\sqrt{d_h}}, \qquad i \in \mathcal{I}.$$

The numerically stable online-softmax state for this shard is

$$\phi = (m, \ell, u),$$

where

$$m = \max_{i\in\mathcal{I}} z_i, \qquad \ell = \sum_{i\in\mathcal{I}} e^{z_i - m}, \qquad u = \sum_{i\in\mathcal{I}} e^{z_i - m} v_i.$$

Its attention response is

$$\mathcal{R}(\phi) = \frac{u}{\ell}.$$

Indeed,

$$\frac{u}{\ell} = \frac{\sum_{i\in\mathcal{I}} e^{z_i - m} v_i}{\sum_{i\in\mathcal{I}} e^{z_i - m}} = \frac{\sum_{i\in\mathcal{I}} e^{z_i} v_i}{\sum_{i\in\mathcal{I}} e^{z_i}},$$

which is the standard softmax attention response over the shard.

B.2 Exact Merge Operator

Consider two disjoint KV shards with index sets $\mathcal{I}_1$ and $\mathcal{I}_2$. For the same receiver probe, let their states be

$$\phi_1 = (m_1, \ell_1, u_1), \qquad \phi_2 = (m_2, \ell_2, u_2).$$

We define

$$\phi_1 \oplus \phi_2 = (m, \ell, u)$$

by

$$\begin{aligned} m &= \max(m_1, m_2), \\ \ell &= e^{m_1 - m}\ell_1 + e^{m_2 - m}\ell_2, \\ u &= e^{m_1 - m}u_1 + e^{m_2 - m}u_2. \end{aligned} \tag{4}$$

The exponential factors express both shard-local states relative to the common normalization reference m. This renormalization introduces no approximation.

B.3 Correctness of the Merge

We show that Equation 4 produces the same response as softmax attention evaluated directly over the union of the two shards.

Let

$$m_j = \max_{i\in\mathcal{I}_j} z_i, \qquad j \in \{1, 2\},$$

and

$$m = \max(m_1, m_2) = \max_{i\in\mathcal{I}_1\cup\mathcal{I}_2} z_i.$$

The denominator of the joint attention computation is

$$\begin{aligned} \sum_{i\in\mathcal{I}_1\cup\mathcal{I}_2} e^{z_i - m} &= \sum_{i\in\mathcal{I}_1} e^{z_i - m} + \sum_{i\in\mathcal{I}_2} e^{z_i - m} \\ &= e^{m_1 - m} \sum_{i\in\mathcal{I}_1} e^{z_i - m_1} + e^{m_2 - m} \sum_{i\in\mathcal{I}_2} e^{z_i - m_2} \\ &= e^{m_1 - m}\ell_1 + e^{m_2 - m}\ell_2 \\ &= \ell. \end{aligned} \tag{5}$$

Similarly, the numerator is

$$\begin{aligned}\sum_{i\in\mathcal{I}_1\cup\mathcal{I}_2} e^{z_i-m}v_i &= \sum_{i\in\mathcal{I}_1} e^{z_i-m}v_i + \sum_{i\in\mathcal{I}_2} e^{z_i-m}v_i \\ &= e^{m_1-m}\sum_{i\in\mathcal{I}_1} e^{z_i-m_1}v_i + e^{m_2-m}\sum_{i\in\mathcal{I}_2} e^{z_i-m_2}v_i \\ &= e^{m_1-m}u_1 + e^{m_2-m}u_2 \\ &= u. \end{aligned} \tag{6}$$

Combining Equations 5 and 6 gives

$$\mathcal{R}(\phi_1 \oplus \phi_2) = \frac{u}{\ell} = \frac{\sum_{i\in\mathcal{I}_1\cup\mathcal{I}_2} e^{z_i-m}v_i}{\sum_{i\in\mathcal{I}_1\cup\mathcal{I}_2} e^{z_i-m}}. \tag{7}$$

Therefore,

$$\mathcal{R}(\phi_1 \oplus \phi_2) = \mathrm{softmax}\left(\frac{q[K_1;K_2]^\top}{\sqrt{d_h}}\right)[V_1;V_2],$$

where $[K_1;K_2]$ and $[V_1;V_2]$ denote concatenation along the KV-token dimension. Thus, for a fixed receiver probe, merging the two online-softmax states exactly recovers the response obtained by evaluating the union of the two KV shards.

B.4 Composition over Multiple Shards

The merge operator extends recursively to any collection of disjoint KV shards. In exact arithmetic, $\oplus$ is commutative and associative:

$$\phi_1 \oplus \phi_2 = \phi_2 \oplus \phi_1,$$

and

$$(\phi_1 \oplus \phi_2) \oplus \phi_3 = \phi_1 \oplus (\phi_2 \oplus \phi_3).$$

Both sides represent the online-softmax state of the same union of KV entries under a common maximum-logit reference.

For receiver probe p, let ϕ_p^L denote the receiver-local state and $\phi_{p,c}$ the state returned for remote candidate shard c. For any selected subset $S \subseteq \mathcal{C}$, define

$$\Phi_p(S) = \phi_p^L \oplus \bigoplus_{c\in S} \phi_{p,c}.$$

The corresponding attention response is

$$A_p(S) = \mathcal{R}(\Phi_p(S)).$$

For the same probe, this is exactly the response obtained by jointly evaluating the receiver-local KV states and the remote KV states contained in S.

The full-candidate reference in Equation 2 is

$$\Phi_p(\mathcal{C}) = \phi_p^L \oplus \bigoplus_{c\in\mathcal{C}} \phi_{p,c},$$

with

$$A_p(\mathcal{C}) = \mathcal{R}(\Phi_p(\mathcal{C})).$$

Both $A_p(S)$ and $A_p(\mathcal{C})$ can therefore be constructed from the same compact per-shard states without transferring the corresponding raw K/V tensors.

B.5 Implication for Pre-Transfer Subset Evaluation

Exact composition allows KVise to evaluate candidates before transferring their KV payloads. Given a fixed query bank $\mathcal{Q} = \{(q_p, w_p)\}_{p=1}^P$, each sender computes one compact online-softmax state for every relevant probe–shard pair. The receiver can then obtain the attention response of a candidate subset by merging only the states associated with that subset.

The distortion

$$D(S) = \sum_{p=1}^{P} w_p \frac{\|A_p(S) - A_p(\mathcal{C})\|_2^2}{\|A_p(\mathcal{C})\|_2^2 + \epsilon}$$

can therefore be evaluated before transferring the raw K/V tensors in S. Candidate payloads are requested only after the selection procedure determines $S^\star$.

The exactness result applies to the attention response of a fixed receiver probe over a specified union of KV states. It does not imply that provisional probe-space distortion equals downstream answer error, nor that future decoding queries remain unchanged after remote KV reuse. As discussed in Appendix A.6, KVise treats $D(S)$ as a pre-transfer surrogate for downstream KV utility, not as a formal task-quality guarantee.

C KV Selection Algorithm and Complexity

After collecting the compact attention states described in Section 2.2 and Appendix B, KVise selects a subset using only candidate metadata and mergeable online-softmax states. It requests the raw candidate K/V tensors only after determining the selected subset.

C.1 Selection Objective

Each candidate shard $c \in \mathcal{C}$ has payload size b_c. The total payload of a subset $S \subseteq \mathcal{C}$ is

$$B(S) = \sum_{c \in S} b_c.$$

Given a distortion threshold τ, KVise seeks

$$S^\star = \arg\min_{S \subseteq \mathcal{C}} B(S) \qquad \text{s.t.} \qquad D(S) \le \tau,$$

where $D(S)$ is defined in Equation 2. By Appendix B, KVise can evaluate $D(S)$ from compact per-shard attention states without transferring their raw K/V payloads.

For each receiver probe p, the full-candidate reference

$$A_p(\mathcal{C}) = \mathcal{R}\left(\phi_p^L \oplus \bigoplus_{c \in \mathcal{C}} \phi_{p,c}\right)$$

is constructed once and reused throughout selection. Candidate subsets are evaluated by changing only which remote states participate in the merge.

C.2 Exact Minimum-Byte Selection

When the candidate space is sufficiently small, KVise evaluates the feasible subset space directly. Let

$$\mathcal{F}_\tau = \{S \subseteq \mathcal{C} \mid D(S) \le \tau\}$$

denote the feasible subsets. The exact branch returns

$$S^\star = \arg\min_{S \in \mathcal{F}_\tau} B(S).$$

For each subset, KVise composes its attention states over all probes, recovers $A_p(S)$, and evaluates $D(S)$. Because this computation uses only compact states returned during pre-transfer evaluation, exhaustive subset search does not repeatedly transmit candidate tensors.

If several feasible subsets have the same minimum payload, the selector returns the first one encountered under the implementation's deterministic ordering. This tie-breaking rule does not alter the minimum-byte criterion.

C.3 Greedy Selection for Larger Candidate Spaces

Direct enumeration becomes expensive as $|\mathcal{C}|$ grows. For larger candidate spaces, KVise constructs the subset iteratively using the conditional distortion reduction per transmitted byte introduced in Section 2.2:

$$\rho(c \mid S) = \frac{D(S) - D(S \cup \{c\})}{b_c}.$$

Starting from

$$S_0 = \varnothing,$$

the selector evaluates every remaining candidate conditioned on the current subset and chooses

$$c_t^\star = \arg\max_{c \in \mathcal{C} \setminus S_t} \rho(c \mid S_t).$$

It then updates the subset as

$$S_{t+1} = S_t \cup \{c_t^\star\}.$$

This process continues until

$$D(S_t) \le \tau.$$

The marginal score is recomputed after every addition. A shard's priority therefore depends on both the receiver probes and the shards already selected, preserving the receiver- and subset-dependent notion of utility used in the main method.

If the selector exhausts all candidates, the resulting set is $\mathcal{C}$. By construction, its distortion relative to the full-candidate reference is zero up to numerical precision. The complete candidate set therefore provides a terminal fallback.

C.4 Complete Selection Procedure

Algorithm 1 summarizes the selection stage. KVise uses exact subset evaluation for small candidate spaces and switches to the iterative marginal-utility procedure when direct enumeration becomes inefficient.

C.5 Computational Complexity

Let

$$C = |\mathcal{C}|$$

be the number of candidate shards, and let P be the number of receiver probes. Candidate evaluation uses compact attention states, so the selector's complexity is independent of the raw K/V tensor size, except for the payload costs $\{b_c\}$ in the objective.

The exact branch evaluates at most

$$2^C$$

candidate subsets. If evaluating the merged response of one subset over all probes costs $\Gamma(P, C)$, the worst-case complexity is

$$O\big(2^C\,\Gamma(P, C)\big).$$

KVise therefore uses this exponential branch only when the candidate space is small enough for direct evaluation.

The iterative branch requires at most C rounds. In the worst case, it performs

$$C + (C-1) + \cdots + 1 = \frac{C(C+1)}{2}$$

conditional candidate evaluations, or

$$O(C^2)$$

evaluations in total. Including attention-state composition over the receiver probes gives the worst-case complexity

$$O\big(C^2\,\Gamma(P, C)\big).$$

Algorithm 1 KVise KV Selection

Require: Candidate set $\mathcal{C}$; payload sizes $\{b_c\}_{c\in\mathcal{C}}$; receiver-local states $\{\phi_p^L\}_{p=1}^P$; remote candidate states $\{\phi_{p,c}\}$; probe weights $\{w_p\}_{p=1}^P$; distortion threshold τ
Ensure: Selected subset $S^\star$

for $p = 1, \ldots, P$ **do**
    Compute

$$A_p(\mathcal{C}) \leftarrow \mathcal{R}\left(\phi_p^L \oplus \bigoplus_{c\in\mathcal{C}} \phi_{p,c}\right)$$

end for
if $|\mathcal{C}|$ is within the exact-search regime **then**
    $S^\star \leftarrow \varnothing$
    $B^\star \leftarrow +\infty$
    for all $S \subseteq \mathcal{C}$ **do**
        Evaluate $D(S)$ using Equation 2
        if $D(S) \le \tau$ **and** $B(S) < B^\star$ **then**
            $S^\star \leftarrow S$
            $B^\star \leftarrow B(S)$
        end if
    end for
    return $S^\star$
else
    $S \leftarrow \varnothing$
    Evaluate $D(S)$
    while $D(S) > \tau$ **do**
        for all $c \in \mathcal{C} \setminus S$ **do**
            Evaluate $D(S \cup \{c\})$

$$\rho(c \mid S) \leftarrow \frac{D(S) - D(S \cup \{c\})}{b_c}$$

        end for

$$c^\star \leftarrow \arg\max_{c\in\mathcal{C}\setminus S} \rho(c \mid S)$$

        $S \leftarrow S \cup \{c^\star\}$
        Evaluate $D(S)$
    end while
    return S
end if

In a straightforward implementation, constructing a subset response is linear in the number of merged shard states and probes. Intermediate merged states can instead be cached within each selection round. Evaluating an extension $S \cup \{c\}$ then requires only composing the current subset state with the candidate state for each probe. Selection consequently operates on compact vectors and scalars rather than repeatedly processing or transmitting complete remote KV tensors.

C.6 Selection Guarantees and Scope

The exact branch returns the minimum-payload subset among all evaluated subsets satisfying

$$D(S) \le \tau.$$

Its optimality is defined with respect to the pre-transfer objective in Equation 1 and the candidate universe $\mathcal{C}$ for the current collaboration episode.

The iterative branch is a scalable heuristic for larger candidate spaces. It prioritizes conditional attention-distortion reduction per transmitted byte. KVise does not assume that $D(S)$ is submodular and does not claim a global approximation guarantee for the greedy solution. The selected

subset should therefore be interpreted as an efficient receiver-conditioned solution to the distortion-controlled allocation problem, not necessarily as the globally minimum-payload subset.

Finally, the stopping condition

$$D(S) \leq \tau$$

applies only to the provisional receiver probe space. As discussed in Appendix A.6, satisfying this constraint does not imply identical downstream generations or task quality relative to Exhaustive Transfer. The threshold τ controls fidelity to the full-candidate attention response used during selection. Its downstream quality–communication trade-off is evaluated empirically in Section 3.2.

D Position Alignment and Receiver-Side Reuse

Selected remote keys must be expressed in the receiver's positional frame before they are inserted into its cache. We describe this alignment and the receiver-side reuse procedure corresponding to Section 2.3. After alignment, local and remote KV states participate in the same joint attention computation during formal replay.

D.1 RoPE-Based Key Re-Expression

For the RoPE-based compatible models used in our implementation, let $k^{(0)}$ denote a content-space key before the rotary positional transformation. If the key is generated at sender-side position p, its stored representation is

$$k_p = R(p)k^{(0)},$$

where $R(p)$ is the RoPE transformation for position p.

To reuse this key at receiver-side position p', KVise removes the sender-side transformation and applies the receiver-side transformation:

$$\widetilde{k}_{p'} = R(p')R(p)^{-1}k_p.$$

Substituting $k_p = R(p)k^{(0)}$ gives

$$\widetilde{k}_{p'} = R(p')R(p)^{-1}R(p)k^{(0)} = R(p')k^{(0)}.$$

Hence, $\widetilde{k}_{p'}$ represents the same key content at receiver position p'.

The corresponding value vector is not position-rotated and remains unchanged:

$$\widetilde{v} = v.$$

Position alignment therefore modifies only the remote key's positional representation.

D.2 Position Mapping

Each selected remote shard includes its sender-side token positions and K/V tensors. During reuse, KVise assigns positions in the receiver replay frame to these remote tokens. Let

$$\pi(p) = p'$$

map sender-side token position p to receiver-side position p'. The aligned key is

$$\widetilde{k}_{\pi(p)} = R(\pi(p))R(p)^{-1}k_p.$$

The mapping $\pi(\cdot)$ is deterministic within a collaboration episode and is fixed before formal replay. Every key in a selected remote shard is transformed according to the same receiver-side positional layout used when the shard is mounted in the receiver cache.

For a selected shard c, let

$$K_c^R = \{k_{p_1}, \ldots, k_{p_n}\}$$

denote its sender-side keys. The aligned receiver-side representation is

$$\widetilde{K}_c^R = \left\{R(\pi(p_i))R(p_i)^{-1}k_{p_i}\right\}_{i=1}^{n}.$$

The corresponding values remain

$$V_c^R = \{v_{p_1}, \ldots, v_{p_n}\}.$$

If a model applies RoPE to only a subset of each head's dimensions, the transformation is restricted to those rotary dimensions. All remaining dimensions are copied unchanged. This procedure preserves the positional convention of the underlying model rather than introducing a separate encoding.

D.3 Layer-Wise Remote KV Construction

After alignment, the selected remote shards are grouped by layer. For layer ℓ, let

$$\widetilde{K}^R_{\ell,S^\star}$$

and

$$V^R_{\ell,S^\star}$$

denote the aligned remote keys and corresponding values from shards in $S^\star$ that contain states for that layer.

Let

$$K^L_\ell, \qquad V^L_\ell$$

denote the receiver-local cache at the same layer. KVise constructs

$$K^\star_\ell = \left[K^L_\ell; \widetilde{K}^R_{\ell,S^\star}\right],$$

and

$$V^\star_\ell = \left[V^L_\ell; V^R_{\ell,S^\star}\right],$$

where concatenation is performed along the KV-token dimension.

This construction does not summarize or fuse independently computed local and remote attention outputs. It creates one augmented KV set over which the receiver performs its native attention operation.

D.4 Joint Attention with Local and Remote KV

Given receiver queries Q_ℓ, attention over the augmented cache is

$$H^\star_\ell = \mathrm{softmax}\left(\frac{Q_\ell (K^\star_\ell)^\top}{\sqrt{d_h}} + M_\ell\right) V^\star_\ell,$$

where M_ℓ is the receiver model's attention mask.

Local and selected remote keys occur inside the same softmax and are normalized jointly. For a receiver query q, the attention mass assigned to a remote state therefore depends on its score relative to both receiver-local keys and other selected remote keys. This is the joint-attention semantics of Equation 3, rather than a post-hoc combination of independently computed attention outputs.

D.5 Stable Cache Restoration and Formal Replay

The suffix pass used to construct the query bank is provisional. Before remote reuse, KVise discards its suffix states and restores the stable local cache saved before probing.

Receiver execution then proceeds as follows:

1. restore the stable local KV cache;
2. position-align the selected remote keys;
3. group the aligned remote K/V states by layer;
4. mount the selected remote states together with the receiver-local cache;
5. replay the same receiver suffix under the augmented KV context; and
6. continue normal autoregressive decoding from the resulting formal receiver state.

Replay is required because the provisional suffix was computed before the selected remote KV states were available. Retaining its hidden states would commit receiver computation that did not incorporate the selected remote context. Restoring the stable cache and replaying the suffix ensures that all formal receiver states used for subsequent decoding are computed under the final augmented KV context.

D.6 Relationship to the Selection Stage

Position alignment and receiver-side reuse begin only after the subset $S^\star$ has been determined. Candidate utility is evaluated from compact attention statistics, without first mounting the corresponding raw K/V tensors at the receiver.

The execution order is

$$\boxed{\begin{array}{c}\text{compact-state evaluation} \;\rightarrow\; \text{selection of } S^\star \\ \rightarrow\; \text{selected KV transfer} \;\rightarrow\; \text{position alignment} \\ \rightarrow\; \text{joint-attention replay}\end{array}}$$

Position alignment therefore does not determine which candidates are available to the selector. It is a reuse-stage operation applied only to raw K/V tensors already selected for communication.

D.7 Scope and Compatibility

The alignment procedure assumes that sender and receiver KV states share the same attention geometry and positional-encoding convention. In the reported experiments, remote KV states are exchanged only between compatible instances within the same model setting.

KVise does not assume that arbitrary heterogeneous model checkpoints can exchange K/V tensors directly. Models with incompatible head geometry, projection spaces, or positional representations would require an additional representation-mapping mechanism before applying the reuse procedure described here.

Within the compatible-model setting studied in this paper, KVise changes neither model parameters nor the model's native attention operation. Selected remote states are re-expressed in the receiver's positional frame, concatenated with the receiver-local cache, and consumed through standard joint attention and autoregressive decoding.