DSSD: Efficient Edge-Device LLM Deployment and Collaborative Inference via Distributed Split Speculative Decoding

Jiahong NING [*1] **Ce ZHENG** [*2] **Tingting YANG** [1 2]

Abstract

Large language models (LLMs) have transformed natural language processing but face critical deployment challenges in device-edge systems due to resource limitations and communication overhead. To address these issues, collaborative frameworks have emerged that combine small language models (SLMs) on devices with LLMs at the edge, using speculative decoding (SD) to improve efficiency. However, existing solutions often trade inference accuracy for latency or suffer from high uplink transmission costs when verifying candidate tokens. In this paper, we propose Distributed Split Speculative Decoding (DSSD), a novel architecture that not only preserves the SLM–LLM split but also partitions the verification phase between the device and edge. In this way, DSSD replaces the uplink transmission of multiple vocabulary distributions with a single downlink transmission, significantly reducing communication latency while maintaining inference quality. Experiments show that our solution outperforms current methods, and codes are at: https://github.com/JasonNing96/DSSD-Efficient-Edge-Computing

1. Introduction

Large language models (LLMs) have revolutionized natural language processing, enabling powerful applications such as conversational agents, machine translation, and code generation (Chen et al., 2024). Despite their capabilities, the LLMs face significant challenges across both devices and cloud. On devices, stringent constraints such as limited memory capacity, restricted battery, and computational power hinder the adoption of traditional LLM frameworks. Cloud deployments, while benefiting from scalable computational resources, suffer from unpredictable network latency. In addition, the mobility of users can lead to frequent connectivity disruptions, which make continuous access to cloud-based services unreliable.

To address these challenges, researchers have proposed a collaborative edge-device architecture that strategically deploys a small language model (SLM) on the device while offloading the large language model (LLM) to a base station (BS) or edge server (Ding et al., 2024; Hao et al., 2024; Shao & Li, 2025). In (Ding et al., 2024), a router trained to predict query difficulty and desired quality level enables cost-efficient assignment of queries to either the small or large model. In (Hao et al., 2024), a cost-aware draft-verification approach was employed. By tuning a predefined threshold p_t for the generated token probability, a controllable performance-cost trade-off was achieved.

However, these studies improve efficiency with a compromise of LLM inference accuracy. Therefore, **speculative decoding (SD)** was taken into account, where a small "draft" model generates γ candidate tokens autoregressive, and then a big "target" model verifies these draft tokens in parallel (Leviathan et al., 2023; Chen et al., 2023). In this way, the inefficiency of autoregressive token generation was mitigated without sacrificing the quality of inference. Furthermore, a **distributed speculative decoding (DSD)** architecture was first introduced in (Zhao et al., 2024) with the draft model for token generation on the device and the target model for verification at the edge or base station (BS). The author tries to optimize the number of tokens generated by SLM to minimize delay and power consumption, taking the uplink and downlink transmission into consideration.

Nevertheless, this hybrid deployment approach is constrained by communication bottlenecks: for each token, the device must transmit a full vocabulary distribution to the BS/edge server for LLM verification, resulting in a communication payload linearly dependent on vocabulary size. In (Oh et al., 2024), the author proposed skipping uplink transmissions and LLM inference on tokens likely to be accepted. This improves token throughput but still at the expense of inference accuracy.

[*]**Equal contribution**. [1]Dalian Maritime University, Dalian, China. [2]Pengcheng Laboratory, Shenzhen, China.
Correspondence to: Ce ZHENG <zhengc@pcl.ac.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Building on previous works, we offer our solution: **a distributed split speculative decoding (DSSD)** framework. Specifically, SLM and LLM are still deployed on device and at edge, separately. But the SD verification phase was further split and distributed across the device and edge. By adopting this method, the uplink transmission of γ vocabulary distributions from SLM is replaced by the downlink transmission of a single vocabulary distribution from LLM, which significantly reduces the communication time and payload.

The rest of the paper is organized as follows: Section 2 introduces the system model and existing DSD system. Section 3 provides our DSSD solutions. Section 4 presents our experimental results and analysis. Conclusions are summarized in Section 5. Appendix A provides the corresponding algorithms.

2. System Model

We consider a collaborative inference architecture between a device and a BS. To achieve efficient collaborative inference, we deploy a lightweight small language model M_q (SLM, denoted as M_q) on the device. Meanwhile, a computationally intensive large language model (LLM, denoted as M_p) is deployed in the BS (Zhao et al., 2024; Oh et al., 2024). We assume a vocabulary $\mathcal{V}$ is shared between SLM and LLM, where $\mathcal{V}$ represents the full set of possible tokens.

2.1. Distributed Speculative Decoding

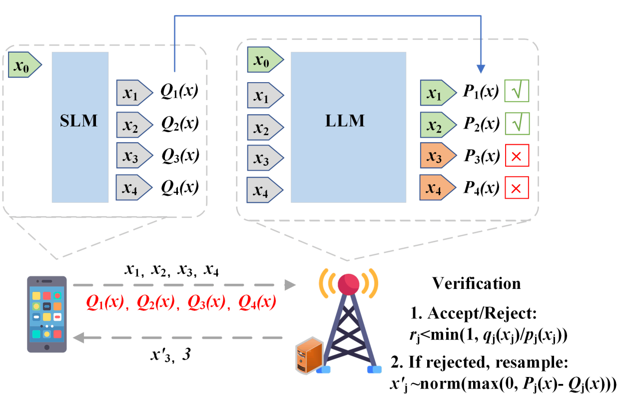

Figure 1. Distributed Speculative Decoding

Fig. 1 illustrates the distributed speculative decoding (DSD) framework: SLM (on device) generates γ draft tokens in an autoregressive manner. And LLM verifies them in parallel, accepting valid tokens and resampling new ones for rejected tokens. A detailed description is presented in Algorithm 1 in the Appendix:

Draft Process (on Deivce): In one single round, SLM generates γ tokens based on the *prefix*. To be more specific, for the i-th token, SLM first gets a vocabulary probability distribution, denoted as $Q_i(x)$ and then samples x_i according to $Q_i(x)$, i.e., $x_i \sim Q_i(x)$.[1]

Uplink Transmission: The device sends the indices of the draft tokens with respect to the vocabulary $\mathcal{V}$ and vocabulary probability distributions to the BS.[2]

Verification Process (at the BS): LLM first gets the $\gamma+1$ distributions according to the prefix and received tokens: $P_j(x), j = 1, \cdots, \gamma+1$. Then it verifies these received tokens: ***a). Accept/Reject:*** Let $q_j(x_j)$ and $p_j(x_j)$ denote the probability values of token x_j in $Q_j(x)$ and $P_j(x)$. If $p_j(x_j) < q_j(x_i)$, x_j is accepted as the j-th token. Otherwise, it is rejected with probability of $1 - q_j(x_j)/p_j(x_j)$.[3] ***b). Resample:*** Once rejected, it resamples a new token $x'_j \sim \text{norm}\left(\max\left(0, P_j(x) - Q_j(x)\right)\right)$. If all γ received tokens are accepted, it samples the $(\gamma+1)$-*th* token $x_{\gamma+1} \sim P_{\gamma+1}(x)$.

Downlink Transmission: BS sends the results x_j and j back to the device, where j denotes the position of the resampled token in the sequence if there is a rejection or equals $\gamma+1$ if all draft tokens are accepted.

2.2. Wireless Communication

To model the end-to-end communication delay, we take into account both the **transmission time (TT)** and the **non-transmission time (NTT)**, the latter comprising processing, propagation, and queuing times. Specifically, the transmission time consists of the uplink transmission time and downlink transmission time:

$$T_{up} = \frac{D_{up}}{R_{up}}, \tag{1}$$

$$T_{down} = \frac{D_{down}}{R_{down}}. \tag{2}$$

where D_{up} and D_{down} are the amount of data transmitted in uplink and downlink, and are R_{up} and R_{down} are the transmission rate.

The round-trip time encompasses the cumulative delays of forward/return propagation, receiver processing, and queuing, which can be treated as a constant.

[1] $Q_i(x)$ or $P_i(x)$ is a vector with the same dimension as the vocabulary, i.e., $|\mathcal{V}|$.

[2] Instead of transmitting full token strings, the device sends only the indices of draft tokens from the vocabulary, reducing communication overhead. For the sake of better illustration, however, tokens are consistently used in Fig. 1 and Fig. 2.

[3] This is equivalent to the part of "Accept" in Algorithm. 1, i.e., $r_j < \min\left\{1, \frac{q_j(x_j)}{p_j(x_j)}\right\}$. The two different representations corresponds to (Leviathan et al., 2023) and (Chen et al., 2023), respectively

Thus, the communication time is

$$T_{comm} = T_{up} + T_{down} + T_{NTT} \tag{3}$$

Given that the index size is insignificant relative to the vocabulary distribution, our analysis considers only the uplink transmission latency associated with the vocabulary distribution. Hence, we have

$$D_{up} = \gamma \cdot |\mathcal{V}| \cdot b_{prob} \tag{4}$$

where $|\cdot|$ denotes the cardinality, b_{prob} represents the bit-width of each probability value, e.g. $b_{prob} = 32$ bits for full precision or 16 bits for half precision.

The communication time becomes

$$T_{comm} = \gamma \cdot \frac{|\mathcal{V}| b_{prob}}{R_{up}} + T_{NTT} \tag{5}$$

2.3. Wall-clock Time

Inference latency comprises three parts: on-device SLM drafting time, edge-side LLM verification time, and device–edge communication time.

In a single run of draft-verify process, the inference latency is:

$$T_{inf} = \gamma \cdot T_{SLM} + T_{comm} + T_{LLM} \tag{6}$$

where T_{LLM} and T_{SLM} denote the time for a single run of M_p and M_q respectively. With (6), we have

$$T_{inf} = \gamma \cdot T_{SLM} + T_{LLM} + \gamma \cdot \frac{|\mathcal{V}| b_{prob}}{R_{up}} + T_{NTT} \tag{7}$$

3. Distributed Split Speculative Decoding

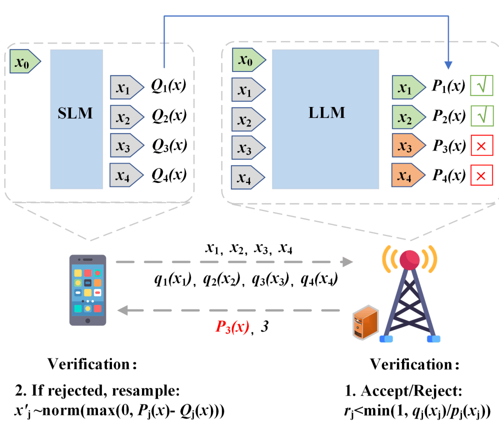

Figure 2. Distributed Split Speculative Decoding.

As shown in Fig. 2, the distributed split speculative decoding (DSSD) employs the same framework as the DSD in Section. 2.1. The only distinction lies in the verification phase, which is split across the device and the edge. Refer to Algorithm 2 for details:

Draft Process: It is exactly the same as DSD in Section. 2.1.

Uplink Transmission: The device sends the indices and the probability values of the draft tokens with respect to the vocabulary $\mathcal{V}$ to the BS.

Verification Process (at the BS): Only the process of ***Accept/Reject*** is still handled by the LLM at the BS. And if all γ received tokens are accepted, it samples the $\gamma + 1$-th token $x_{\gamma+1} \sim P_{\gamma+1}(x)$.

Downlink Transmission: If there is a rejection, BS sends the vocabulary distribution $P_j(x)$ and j back to the device, where j denotes the position of the resampled token in the sequence. Otherwise (all draft tokens are accepted), it sends the $(\gamma + 1)$-*th* token along with the index $\gamma + 1$.

Verfication Process (on Device): the process of ***Resample*** is done on device if there is a rejection, i.e., the received $j < \gamma + 1$.

3.1. Wall-clock Time

In DSSD, data transmission is primarily dominated by downlink traffic. The downlink transmission of vocabulary distribution occurs only when a rejection happens. Accordingly, the downlink transmission time can be expressed as

$$T_{down} = a \cdot \frac{|\mathcal{V}| b_{prob}}{R_{down}} \tag{8}$$

where $a = 1$ if there is a rejection, otherwise $a = 0$.

As the probability of all draft tokens being accepted is α^{γ}, the average communication time is

$$T_{comm} = (1 - \alpha^{\gamma}) \cdot \frac{|\mathcal{V}| b_{prob}}{R_{down}} + T_{NTT} \tag{9}$$

Hence, we have

Corollary 3.1. *The communication time is bounded by*

$$T_{NTT} \leq T_{comm} \leq T_{down} + T_{NTT} \tag{10}$$

As $\gamma \to \infty$, the probability of at least one rejection tends to 1. For sufficiently large γ, the communication time could be approximated as

$$T_{comm} \approx T_{down} + T_{NTT} \tag{11}$$

On the other hand, as $\alpha \to 1$, all tokens will be accepted. And.

$$T_{comm} \approx T_{NTT} \tag{12}$$

γ \ α	0.5	0.6	0.7	0.8	0.9	0.99
2	0.75 (26.00 ms)	0.64 (25.12 ms)	0.51 (24.08 ms)	0.36 (22.88 ms)	0.19 (21.52 ms)	0.02 (20.16 ms)
4	0.94 (27.50 ms)	0.87 (26.96 ms)	0.76 (26.08 ms)	0.59 (24.72 ms)	0.34 (22.75 ms)	0.04 (20.32 ms)
6	0.98 (27.88 ms)	0.95 (27.63 ms)	0.88 (27.06 ms)	0.74 (25.90 ms)	0.47 (23.75 ms)	0.06 (20.47 ms)
8	1.00 (27.97 ms)	0.98 (27.87 ms)	0.94 (27.54 ms)	0.83 (26.66 ms)	0.57 (24.56 ms)	0.08 (20.62 ms)

Table 1. The value of $1-\alpha^{\gamma}$ and expected communication time T_{comm}.

Table. 1 shows the value of $1-\alpha^{\gamma}$ and communication time under different α and γ.

The inference latency is further expressed as:

$$T_{inf}=\gamma\cdot T_{SLM}+T_{LLM}+(1-\alpha^{\gamma})\cdot\frac{|\mathcal{V}|b_{prob}}{R_{down}}+T_{NTT} \quad (13)$$

4. Experiment

To assess the proposed "Distributed Split Speculative Decoding" framework, we designed and employed an end-to-end experimental setup that simulates the computational heterogeneity of devices and the base station, as well as the network conditions between them. Our main evaluation metric is the ***Speedup ratio***, which quantifies how much the throughput improves when using the existing DSD or proposed DSSD method compared to conventional LLM inference. That is

$$Speedup=\frac{TP_{DSD}}{TP_{LLM}} \quad (14)$$

or

$$Speedup=\frac{TP_{DSSD}}{TP_{LLM}} \quad (15)$$

where TP_{DSD}, TP_{DSSD} and TP_{LLM} are the throughputs of DSD, DSSD and LLM, respectively.

In this setup, two types of GPUs are configured to emulate the computational heterogeneity. On the device side, a lightweight draft model (OPT-125M) generates speculative draft tokens, while the base station employs a larger target model (OPT-13B or OPT-6.7B) to verify and correct any token prediction errors. Controllable latency and throughput limits were introduced into the communication between the device and BS to simulate real link conditions. The round-trip latency was configured as one of the values in $\{0, 20, 50\}$ ms, and the bandwidth capped at 100 Mbps (Affandi et al., 2024; Ateya et al., 2018).

Each experiment begins with a common 128-token narrative prompt and generates an additional 128 tokens using the top-k sampling with a temperature of 1.0 and $k=10$, in order to maintain full distribution fidelity.

4.1. Homogeneous device

In a homogeneous experiment, two H800 chips are deployed. One runs a small OPT-125M draft model on the device side[4], while the other runs a large OPT-13B or OPT-6.7B verification model on the base station side. This setup isolates the effect of heterogeneous devices on speculative decoding throughput while still accounting for the latency introduced by the wireless environment. Table 2 presents the expected acceleration speed, predicted by our analytical model, and the empirical acceleration speed measured end-to-end on homogeneous hardware. Table 2 presents the expected acceleration speed, predicted by our analytical model, and the empirical acceleration speed measured end-to-end on homogeneous hardware.

As shown in the table.2, the communication time of the DSSD method is much less than that of DSD. The amount of data uploaded by the DSD method is approximately 61,269 bytes per round, in contrast to the DSSD method, which uploads less than 50 bytes each time. This is because the DSSD method converts the draft model's "overloaded" uplink transmission into the target model's "minimal" downlink transmission, which significantly reduces the communication load. As a result, the DSSD method achieves a SPEED much greater than DSD, thus delivering higher overall inference performance. Specifically, the speed-up of the DSSD method is between $1.5\times$ and $2.4\times$, while the speed-up of DSD only remains at $1\times$ or even lower. This is the direct result after fully considering the communication cost.

When the draft length is fixed at $\gamma=8$ for the OPT-125M→6.7B pair, the Speedup ratio shows significant variation for DSD, in contrast to the relatively consistent performance of DSSD. While for DSSD with different targe models, i.e., OPT-125M→6.7B and OPT-125M→13B, the Speedup ratio declines with a lower acceptance rate, primarily due to increased misalignment resulting from the larger model size.

[4]Assume the device operates under limited memory and computational capacity.

NTT (ms)	BW (Mbps)	γ	OPT-125M→6.7B DSD($\alpha = 0.61 \pm 0.05$)		OPT-125M→6.7B DSSD($\alpha = 0.61 \pm 0.05$)		OPT-125M→13B DSSD($\alpha = 0.53 \pm 0.05$)	
			T_{com}(s)	Speedup	T_{com}(s)	Speedup	T_{com}(s)	Speedup
0	100	8	3.91	1.51	0.01	2.28	1.40	1.67
20	100	8	4.60	1.41	0.62	2.31	9.21	1.72
50	10	8	41.44	0.43	1.55	2.19	2.30	1.62
20	50	8	8.59	1.24	0.62	2.22	0.92	1.77
50	50	8	9.52	1.08	1.55	1.97	2.30	1.70
50	10	6	8.12	1.16	0.40	1.57	2.01	1.97

Table 2. Homogeneous experiment.

4.2. Heterogeneous device

In heterogeneous experiments, we deployed a small OPT-125M draft model on an NVIDIA A6000 GPU. Meanwhile, the large OPT-13B or OPT-6.7B verification model ran on an NVIDIA H800 GPU. As DSSD has demonstrated superior performance over DSD in the preceding experiments, our comparisons are restricted to variations in the target LLM's model size.

To better analyze the experimental results, we first introduce and compute the ratio $c = T_{SLM}/T_{LLM}$ as defined in (Leviathan et al., 2023). For the 125M→6.7B configuration, c is approximately 0.1, while for 125M→13B, c is about 0.05. As a result, the SLM's multiple executions are masked by a single LLM decoding. Fixing the draft length at $\gamma = 6$ and varying the link conditions shows that an ideal $0\,\text{ms}/100\,\text{Mbps}$ channel yields a modest $1.02\times$ speedup for the OPT-6.7B target model, and a larger $1.48\times$ for OPT-13B. With a 20ms NTT at the same bandwidth, the advantage shifts to the larger model, whose throughput peaks at 1.75, whereas the smaller model's throughput drops to 0.92. When bandwidth is throttled to $50\,\text{Mbps}$ under the same latency, the trend reverses: OPT-6.7 B rebounds to $1.38\times$, whereas OPT-13 B falls to $1.24\times$, underscoring the latter's bandwidth sensitivity.

Holding the link at $20\,\text{ms}/100\,\text{Mbps}$ while varying the draft length reveals an interior optimum. A short draft ($\gamma = 4$) does not sufficiently amortize the communication delay, limiting speedups to $0.82\times$ for 6.7B and $1.37\times$ for 13B. Extending the draft to $\gamma = 6$ balances synchronization costs and speculative overhead, achieving global maxima of $0.92\times$ and $1.75\times$. Increasing the draft further to $\gamma = 8$ overloads the speculative path, reducing speedups to $0.87\times$ and $1.14\times$. Overall, these results demonstrate that even on an asymmetric A6000–H800 pair, speculative decoding remains effective: with proper tuning of the draft length and moderate network resources, throughput gains of up to $1.8\times$ for OPT-13B and $1.4\times$ for OPT-6.7B can be sustained under realistic communication conditions.

NTT (ms)	BW (Mbps)	γ	6.7B(DSD)	125M→6.7B	125M→13B
			Speedup	Speedup	Speedup
0	100	8	0.88	0.98	1.17
20	100	8	0.85	0.92	1.13
20	50	8	0.72	0.92	1.12
20	100	4	1.10	1.11	1.44
20	100	6	1.12	1.14	1.58
20	100	8	0.93	1.04	1.30

Table 3. Heterogeneous experiment.

5. Conclusion

This work presents an early but meaningful exploration of distributed LLM inference using speculative decoding. Under the current settings, communication time is the main bottleneck for throughput. To reduce communication overhead while maintaining inference performance, the DDSD solution was proposed: the verification process is divided into two stages—"Accept/Reject" and "Resample"—which are executed at the BS and on the device, respectively. In this way, the uplink transmission of γ probability distribution was reduced to γ token index and probability values, and the downlink transmission of one probability distribution at most. Experimental evaluations provide empirical support for these design choices and offer insights into throughput behavior under different draft length and communication conditions.

Acknowledgement

This work was supported by the National Key Research and Development Program of China under Grant No.2024YFE0200800. The methods presented in this work are covered by a filed Patent with Application No.CN2025108856278.

References

Affandi, M. A., Riyadi, M. A., and Prakoso, T. Throughput and coverage evaluation on the use of existing cellular towers for 5g network in surakarta city. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)*, 10 (1):54–72, 2024.

Ateya, A. A., Muthanna, A., Makolkina, M., and Koucheryavy, A. Study of 5g services standardization: Specifications and requirements. In *2018 10th international congress on ultra modern telecommunications and control systems and workshops (ICUMT)*, pp. 1–6. IEEE, 2018.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Chen, Z., Yang, H. H., Chong, K. F. E., and Quek, T. Q. S. Personalizing semantic communication: A foundation model approach. In *2024 IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, pp. 846–850. IEEE, 2024.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L. V., and Awadallah, A. H. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

Hao, Z., Jiang, H., Jiang, S., Ren, J., and Cao, T. Hybrid slm and llm for edge-cloud collaborative inference. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, pp. 36–41, 2024.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Oh, S., Kim, J., Park, J., Ko, S.-W., Quek, T. Q., and Kim, S.-L. Uncertainty-aware hybrid inference with on-device small and remote large language models. *arXiv preprint arXiv:2412.12687*, 2024.

Shao, J. and Li, X. Ai flow at the network edge. *IEEE Network*, 2025.

Zhao, W., Jing, W., Lu, Z., and Wen, X. Edge and terminal cooperation enabled llm deployment optimization in wireless network. In *2024 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, pp. 220–225. IEEE, 2024.

A. Algorithms

Algorithm 1 gives the full details of Distributed Speculative Decoding (DSD). And Algorithms 2 gives the full details of Distributed Split Speculative Decoding (DSSD).

Algorithm 1 Distributed Speculative Decoding (DSD)

▷ Draft Process
Device: Initialize $\mathbf{y} = []$
  Input: prefix, SLM – M_q, token length – γ
  ▷ Sample γ tokens autoregressively
  for $i = 1$ **to** γ **do**
    $Q_i(x) \leftarrow M_q(prefix + \mathbf{y})$;
    Sample $x_i \sim Q_i(x)$;
    $\mathbf{y} = \mathbf{y} + x_i$
  end for
  Output: $[x_1, \cdots, x_\gamma]$ and $[Q_1(x), \cdots, Q_\gamma(x)]$

▷ Uplink Transmission
  Send tokens $[x_1, \cdots, x_\gamma]$ and probability distributions $[Q_1(x), \cdots, Q_\gamma(x)]$ to the BS via uplink transmission[5]

▷ Verification Process
Edge: Initialize $j = 1$, Flag $= 1$
  Input: prefix, LLM – M_p
  ▷ Run M_p in parallel
  $P_1(x), \cdots, P_{\gamma+1}(x) \leftarrow M_q(prefix), \cdots, M_q(prefix + [x_1, \cdots, x_\gamma])$
  while $j \leq \gamma$ & Flag $= 1$ **do**
    Sample $r_j \sim U[0, 1]$ from a uniform distribution.
    ▷ Accept
    if $r_j < \min\left\{1, \frac{q_j(x_j)}{p_j(x_j)}\right\}$ **then**
      x_j is accepted;
      $j = j + 1$
    ▷ Reject and Resample
    else
      $x'_j \sim \text{norm}\left(\max\left(0, P_j(x) - Q_j(x)\right)\right)$;
      $x_j = x'_j$;
      Flag $= 0$
    end if
    if $j = \gamma + 1$ **then**
      Sample $x_{\gamma+1} \sim P_{\gamma+1}(x)$
    end if
  end while

▷ Downlink Transmission
  Send x_j and j

▷ Reset
  Let prefix $=$ prefix $+ [x_1, \cdots, x_j]$ for Device and Edge

[5]The device actually sends the indices of the draft tokens $[x_1, \cdots, x_\gamma]$ from the Vocabulary $\mathcal{V}$.

Algorithm 2 Distributed Split Speculative Decoding (DSSD)

▷ Draft Process
Device: Initialize $\mathbf{y} = []$
  Input: prefix, SLM – M_q, token length – γ
  ▷ Sample γ tokens autoregressively
  for $i = 1$ **to** γ **do**
    $Q_i(x) \leftarrow M_q(prefix + \mathbf{y})$;
    Sample $x_i \sim Q_i(x)$;
    $\mathbf{y} = \mathbf{y} + x_i$
  end for
  Output: $[x_1, \cdots, x_\gamma]$ and $[Q_1(x), \cdots, Q_\gamma(x)]$

▷ Uplink Transmission
  Send tokens $[x_1, \cdots, x_\gamma]$ and probability values $[q_1(x), \cdots, q_\gamma(x)]$ to the BS via uplink transmission

▷ Verification Process
Edge: Initialize $j = 1$, Flag $= 1$
  Input: prefix, LLM – M_p
  ▷ Run M_p in parallel
  $P_1(x), \cdots, P_{\gamma+1}(x) \leftarrow M_q(prefix), \cdots, M_q(prefix + [x_1, \cdots, x_\gamma])$
  while $j \leq \gamma$ & Flag $= 1$ **do**
    Sample $r_j \sim U[0, 1]$ from a uniform distribution.
    ▷ Accept
    if $r_j < \min\left\{1, \frac{q_j(x_j)}{p_j(x_j)}\right\}$ **then**
      x_j is accepted;
      $j = j + 1$
    ▷ Reject
    else
      Flag $= 0$
    end if
    if $j = \gamma + 1$ **then**
      Sample $x_{\gamma+1} \sim P_{\gamma+1}(x)$
    end if
  end while

▷ Downlink Transmission
  Send $P_j(x)$ and j if Flag $= 0$ or x_j and j if Flag $= 1$

▷ Resample
Device:
  if Flag $= 0$ **then**
    $x'_j \sim \text{norm}\left(\max\left(0, P_j(x) - Q_j(x)\right)\right)$;
    $x_j = x'_j$;
  end if

▷ Reset
  Let prefix $=$ prefix $+ [x_1, \cdots, x_j]$ for Device and Edge.
  Note: the device should still need to upload the resampled x_j to edge if Flag $= 0$. But this can be done in the next round of draft-verify process