FedRDMA: Communication-Efficient Cross-Silo Federated LLM via Chunked RDMA Transmission

Zeling Zhang, **Dongqi Cai**, Yiran Zhang, Mengwei Xu,
Shangguang Wang, Ao Zhou

Beijing University of Posts and Telecommunications (BUPT)
Beiyou Shenzhen Institute

The 4rd Workshop on Machine Learning and Systems, co-located with EuroSys 2024.

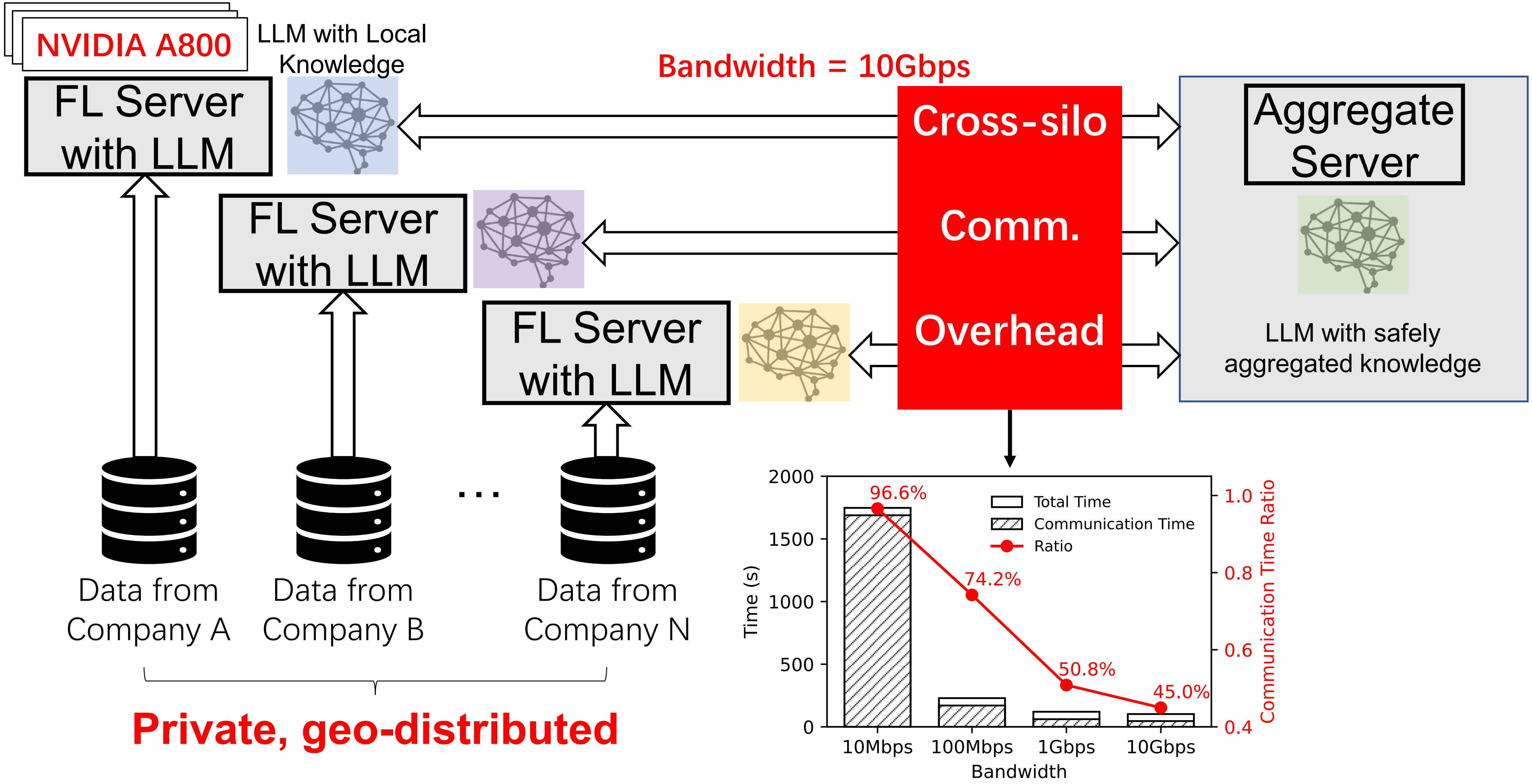
Background

Cross-silo FedLLM

&

Communication-Efficient RDMA

Cross-silo FedLLM



NVIDIA A800

LLM with Local Knowledge

Bandwidth = 10Gbps

FL Server with LLM

FL Server with LLM

FL Server with LLM

Cross-silo Comm. Overhead

Aggregate Server

LLM with safely aggregated knowledge

Data from Company A

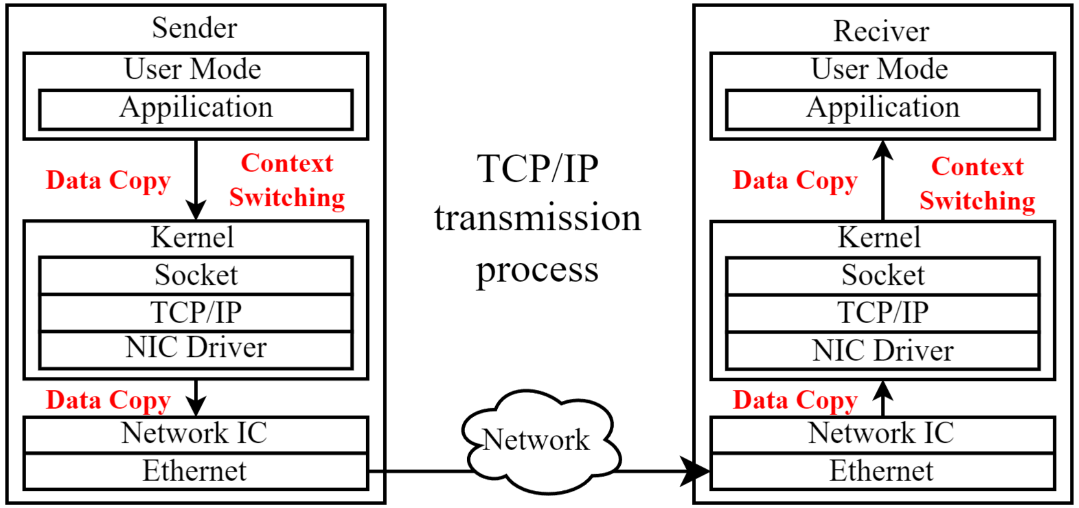
Data from Company B

...

Data from Company N

Private, geo-distributed

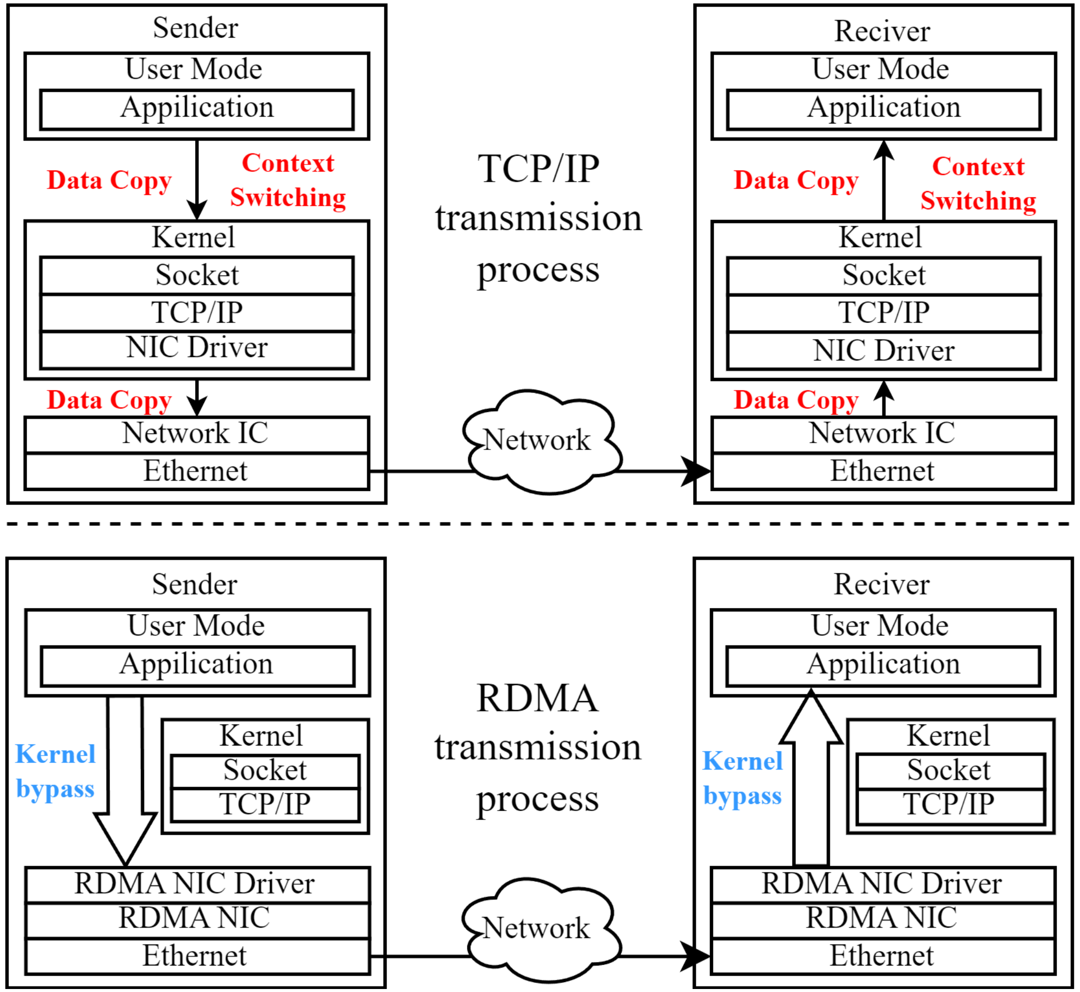
Comunication-Efficient RDMA

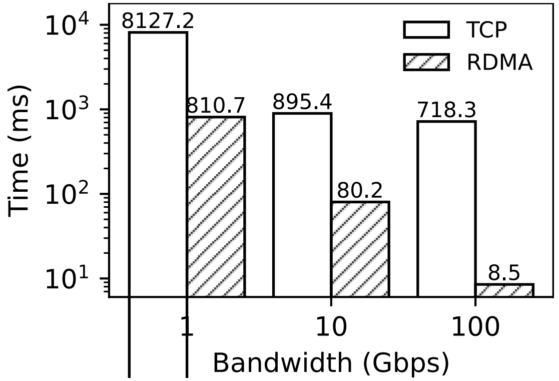


Challenges:

1. FedLLM communication overhead

Comunication-Efficient RDMA



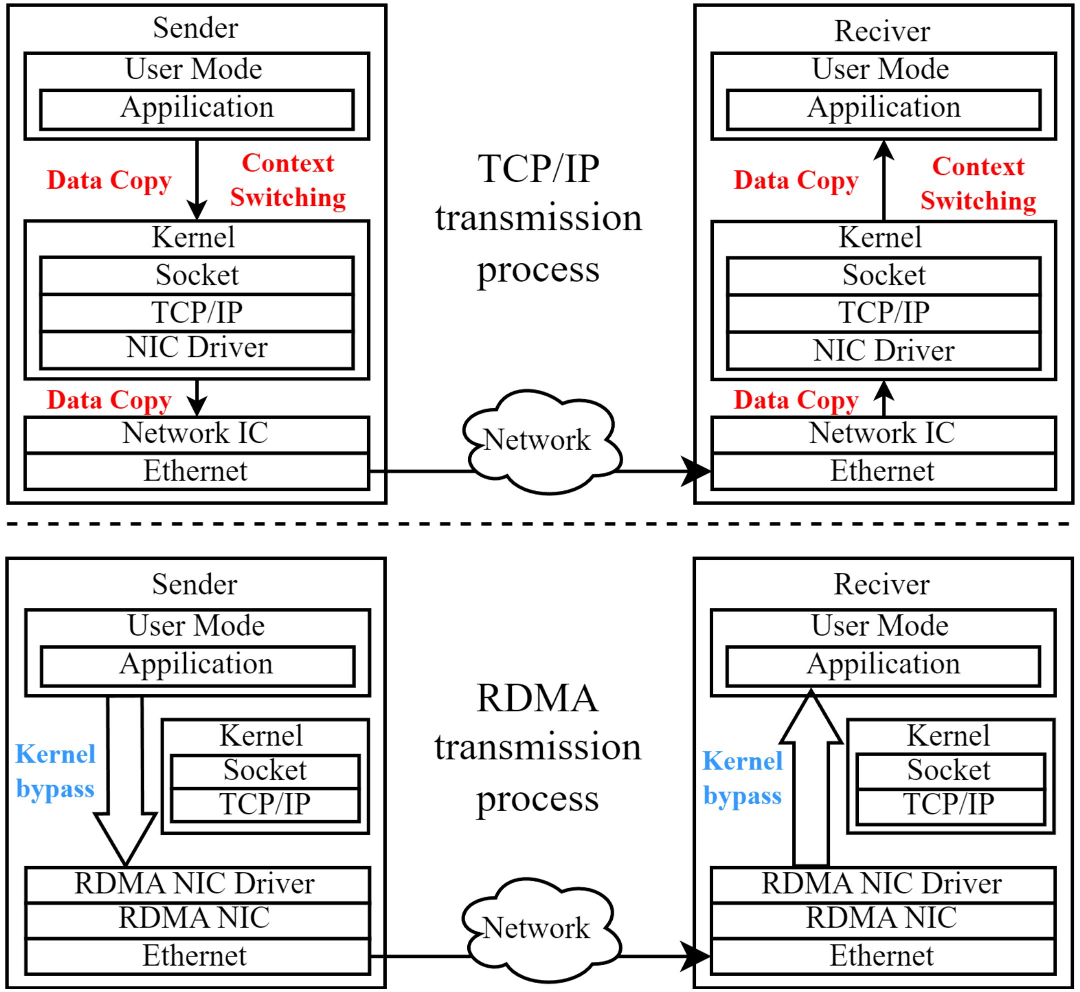


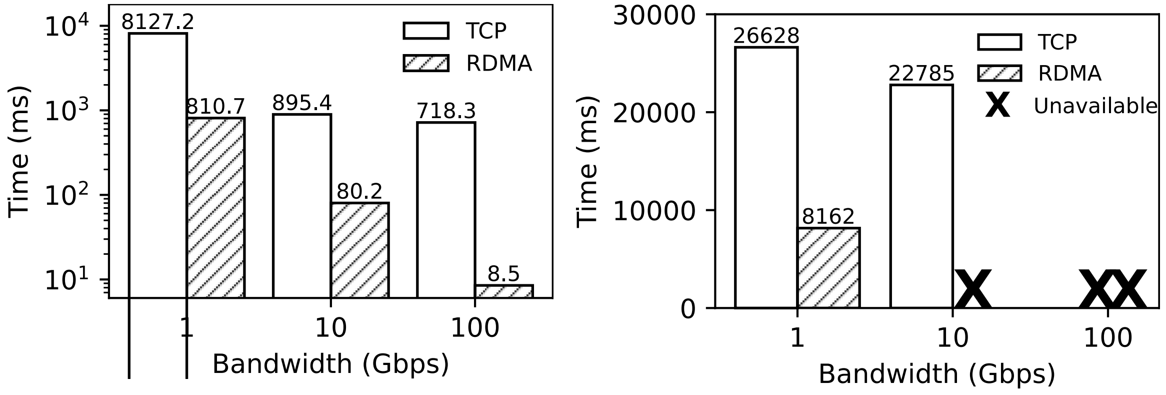
(a) In-Domain (LAN)

Challenges:

1. FedLLM communication overhead

Comunication-Efficient RDMA





(a) In-Domain (LAN)

(b) Cross-Domain (WAN)

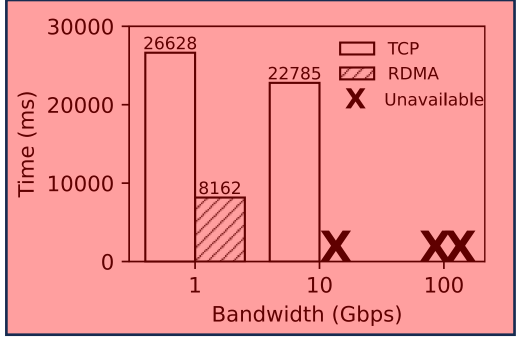
Challenges:

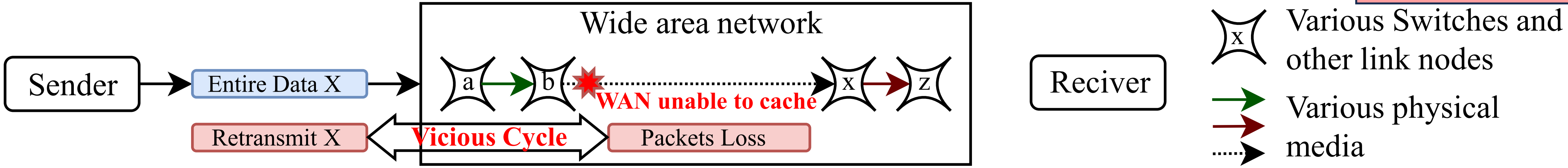
1. FedLLM communication overhead
2. RDMA fails in WAN

Our system:

FedRDMA

Why RDMA Fails





Sender

Entire Data X

Wide area network

WAN unable to cache

Retransmit X

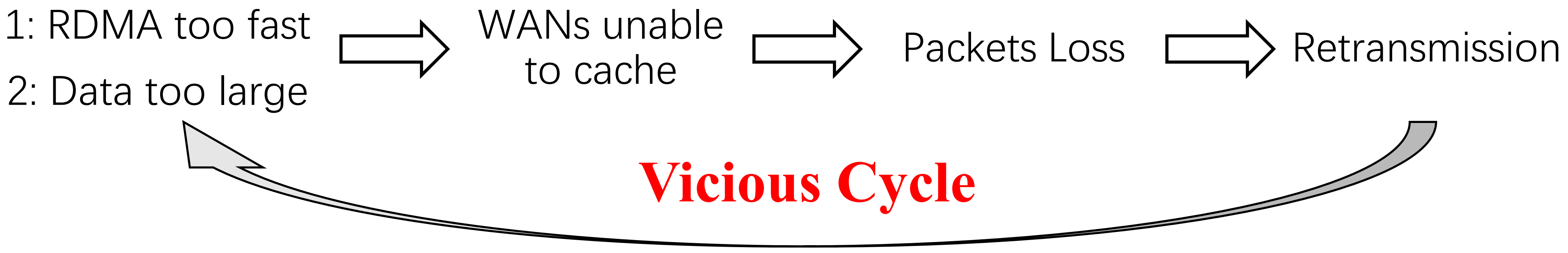
Vicious Cycle

Packets Loss

Reciver

Various Switches and other link nodes

Various physical media



1: RDMA too fast

2: Data too large

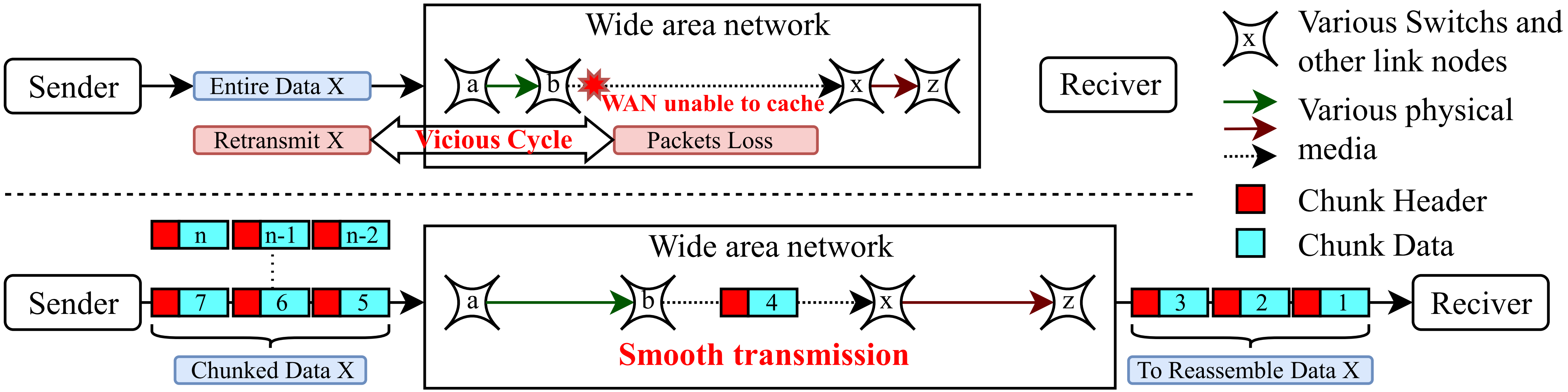
WANs unable to cache

Packets Loss

Retransmission

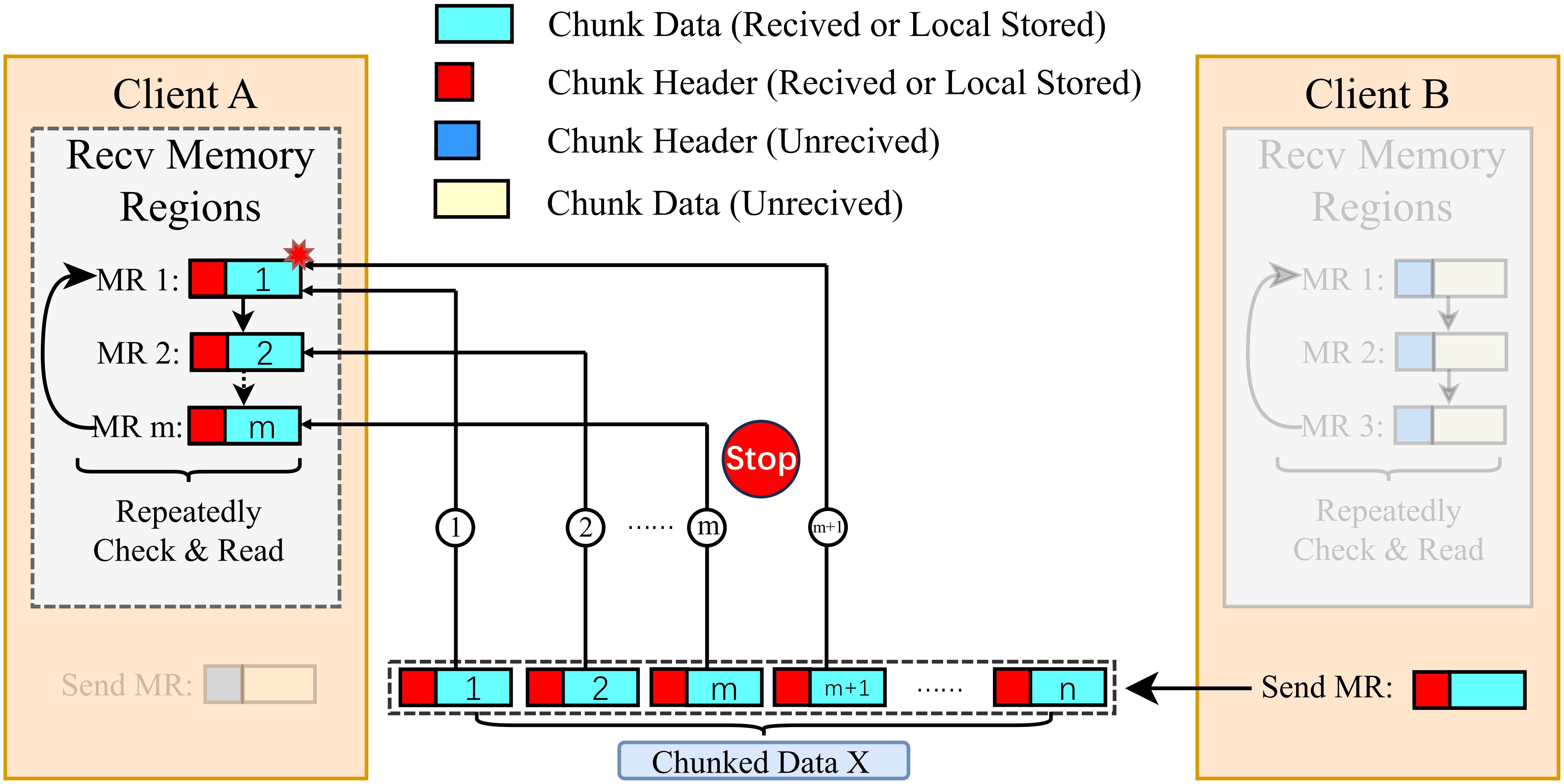
Vicious Cycle

FedRDMA



Split X into n chunks ⇨ Send sequentially with ACKs ⇨ Parse and reassemble

Challenges of FedRDMA



🟦 Chunk Data (Recived or Local Stored)

🟥 Chunk Header (Recived or Local Stored)

🟦 Chunk Header (Unrecived)

🟨 Chunk Data (Unrecived)

Client A

Recv Memory Regions

MR 1: 1

MR 2: 2

MR m: m

Repeatedly Check & Read

Send MR:

Stop

1

2

......

m

m+1

1 2 m m+1 ...... n

Chunked Data X

Client B

Recv Memory Regions

MR 1:

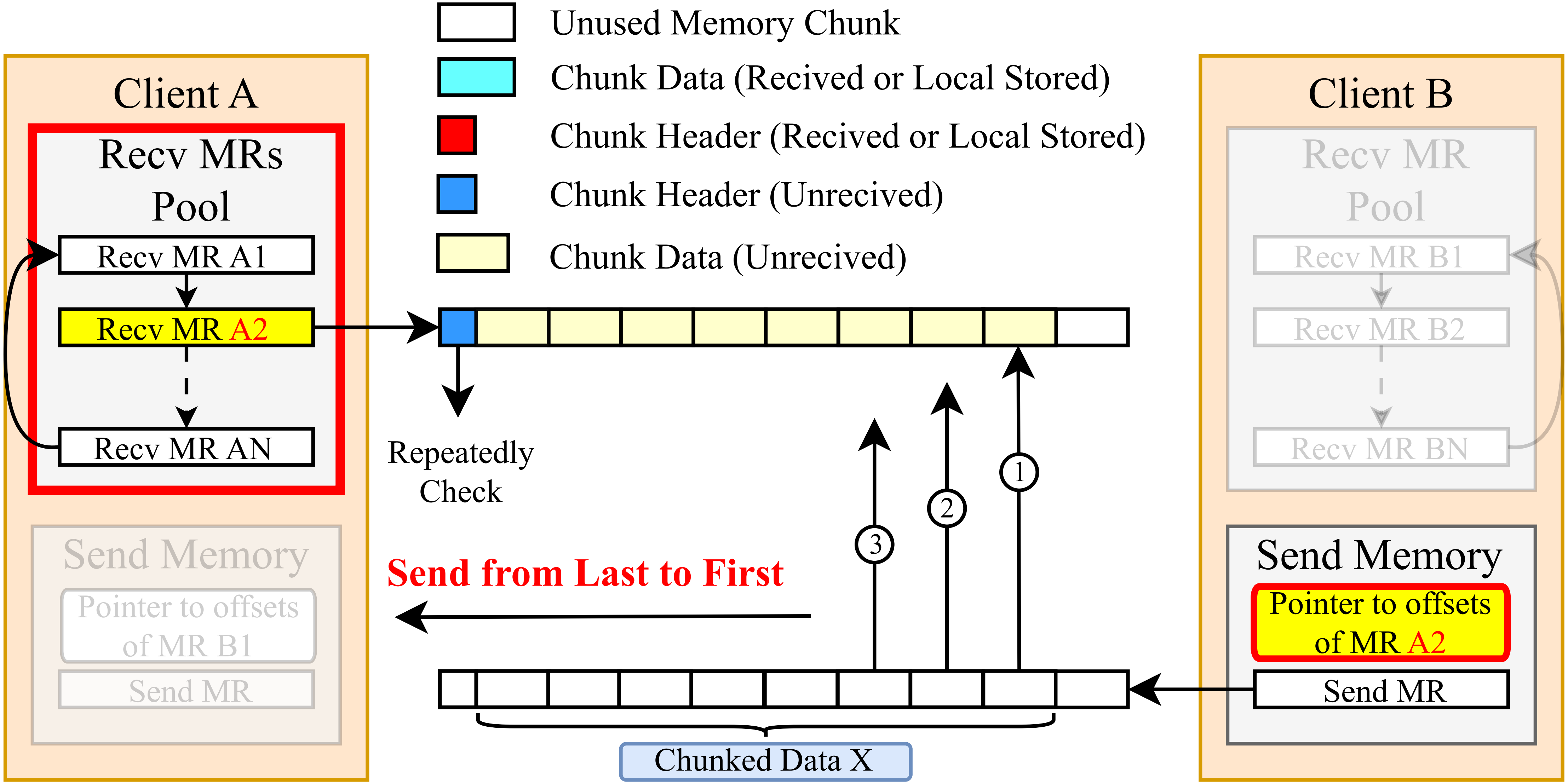
MR 2:

MR 3:

Repeatedly Check & Read

Send MR:

Optimizations of FedRDMA-E



□ Unused Memory Chunk

🟦 Chunk Data (Recived or Local Stored)

🟥 Chunk Header (Recived or Local Stored)

🟦 Chunk Header (Unrecived)

🟨 Chunk Data (Unrecived)

Client A

Recv MRs Pool

Recv MR A1

Recv MR A2

Recv MR AN

Send Memory

Pointer to offsets of MR B1

Send MR

Repeatedly Check

Send from Last to First

3

2

1

Chunked Data X

Client B

Recv MR Pool

Recv MR B1

Recv MR B2

Recv MR BN

Send Memory

Pointer to offsets of MR A2

Send MR

Evaluation

End to end performance and system cost

Experiment setup

Dataset: AdvertiseGen
Model: GPT-2

Baselines:

1. Vanilla FedLLM
2. FedRDMA
3. FedRDMA-E

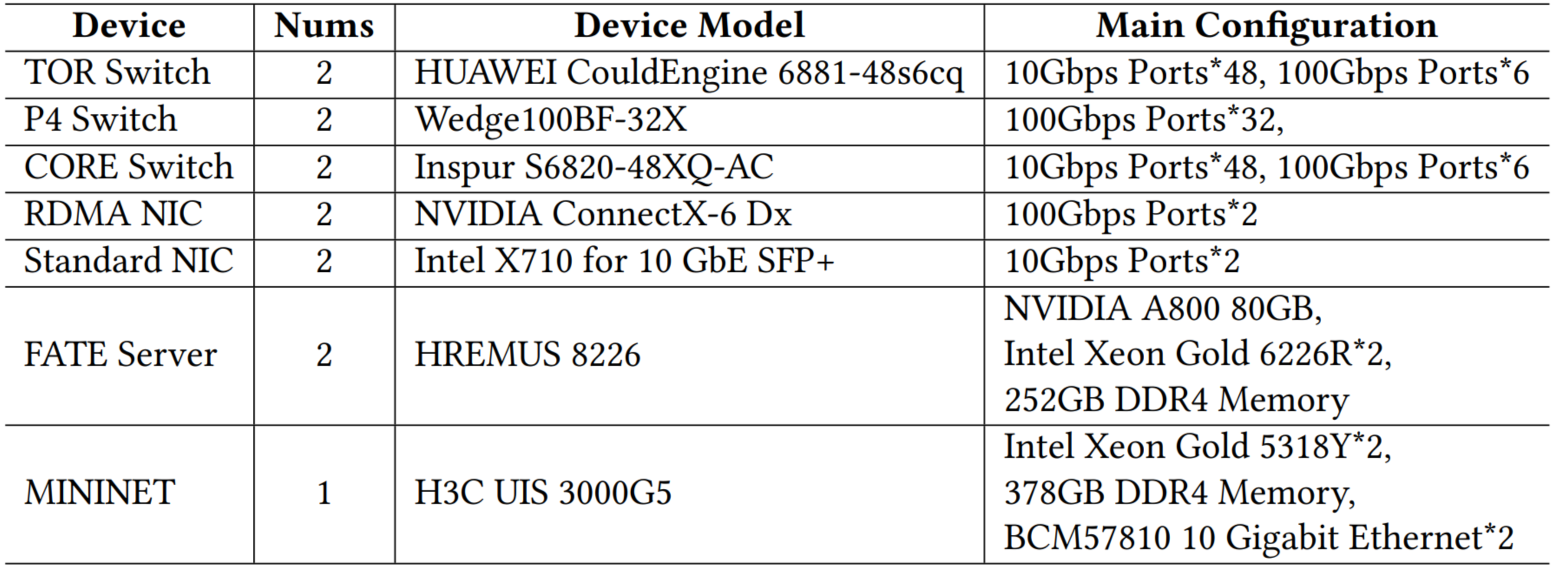
Hyperparameters:

1. WAN RTT time: 20ms
2. Chunk size: 4MB

Software:

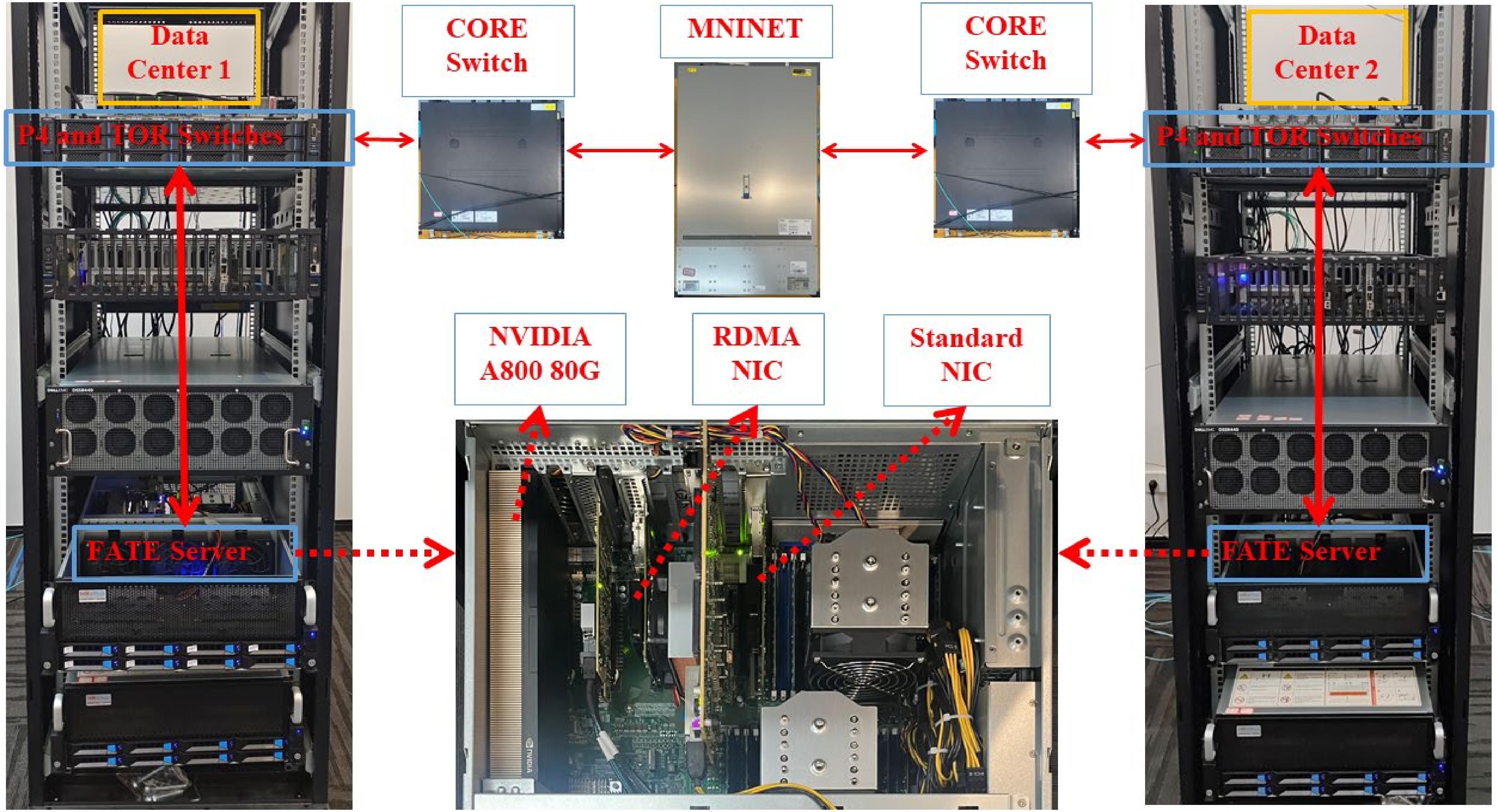
1. FATE-LLM-1.3.0 atop FATE 1.11.3
2. RDMA-CORE 37.4

Hardware:

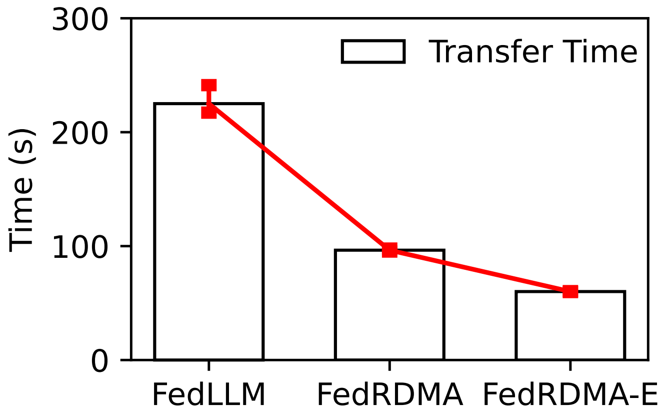


Device	Nums	Device Model	Main Configuration
TOR Switch	2	HUAWEI CouldEngine 6881-48s6cq	10Gbps Ports*48, 100Gbps Ports*6
P4 Switch	2	Wedge100BF-32X	100Gbps Ports*32,
CORE Switch	2	Inspur S6820-48XQ-AC	10Gbps Ports*48, 100Gbps Ports*6
RDMA NIC	2	NVIDIA ConnectX-6 Dx	100Gbps Ports*2
Standard NIC	2	Intel X710 for 10 GbE SFP+	10Gbps Ports*2
FATE Server	2	HREMUS 8226	NVIDIA A800 80GB, Intel Xeon Gold 6226R*2, 252GB DDR4 Memory
MININET	1	H3C UIS 3000G5	Intel Xeon Gold 5318Y*2, 378GB DDR4 Memory, BCM57810 10 Gigabit Ethernet*2

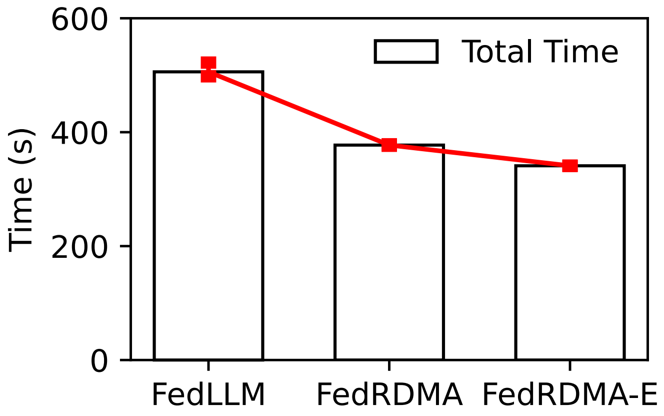
Physical layout and network topology



End-to-end Performance



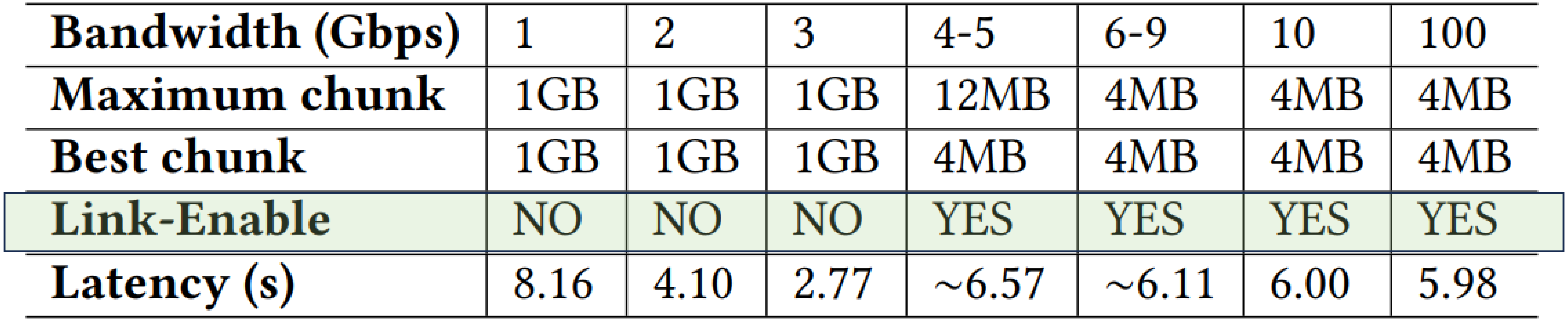
(a) Communication time



(b) End to end time

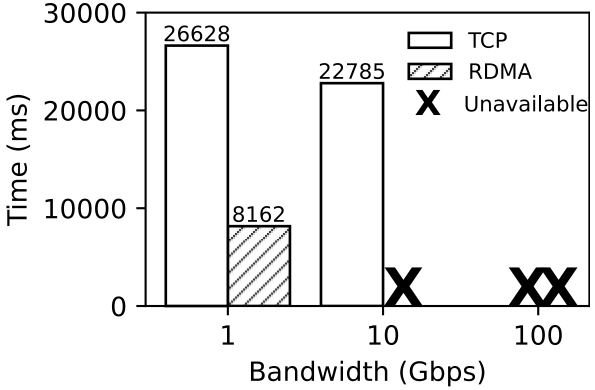
- FedRDMA was able to reduce end-to-end communication time by 73.9%.
- FedRDMA-E ultimately result in a 33.3% reduction in overall end-to-end federated learning time.

Impact of Different Hyperparameters



Bandwidth (Gbps)	1	2	3	4-5	6-9	10	100
Maximum chunk	1GB	1GB	1GB	12MB	4MB	4MB	4MB
Best chunk	1GB	1GB	1GB	4MB	4MB	4MB	4MB
Link-Enable	NO	NO	NO	YES	YES	YES	YES
Latency (s)	8.16	4.10	2.77	~6.57	~6.11	6.00	5.98

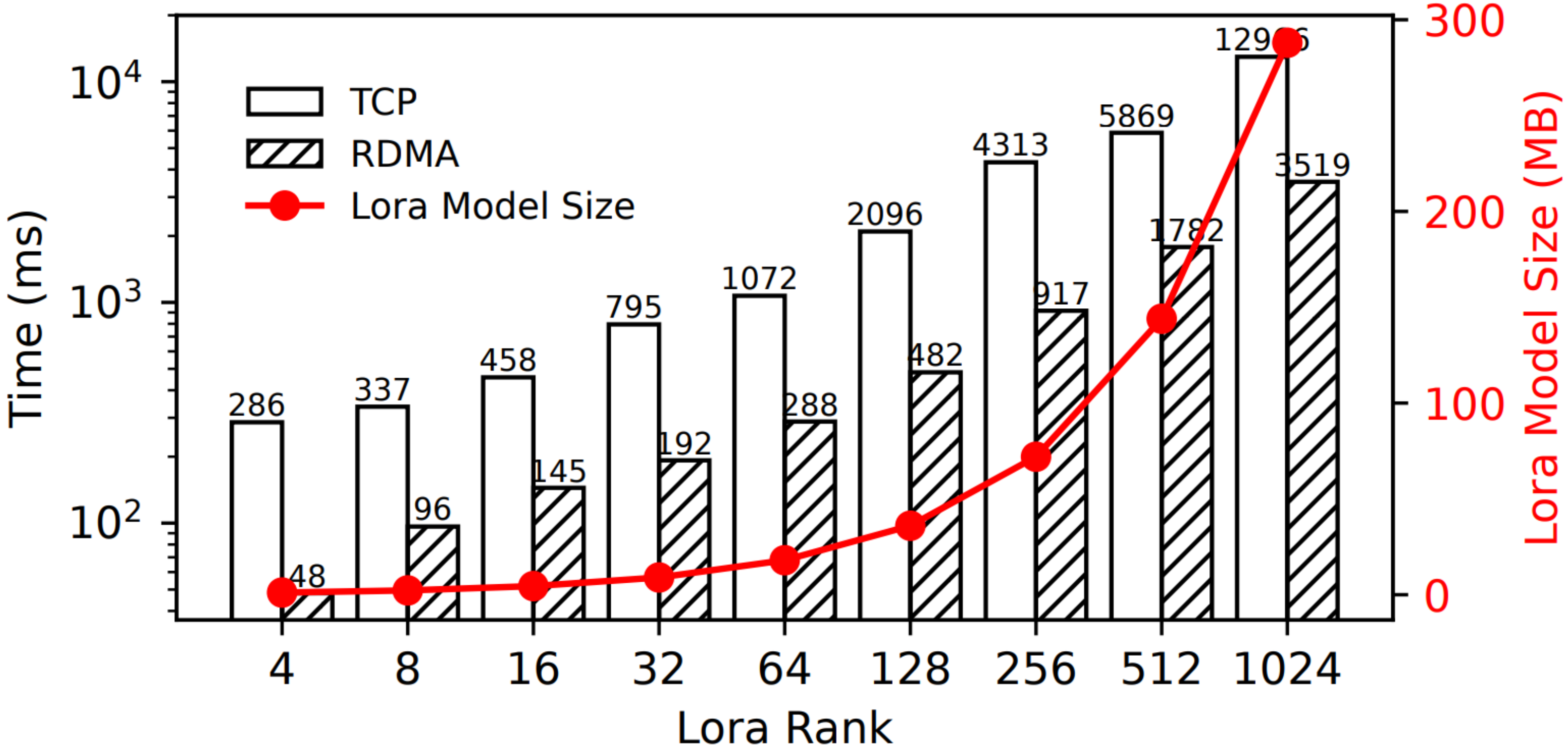
- FedRDMA continuously outperforms TCP a lot from 1Gbps to 100Gbps.
- Link-Enable (send a smaller data chunk first) is needed at higher RDMA bandwidths.



(b) Cross-Domain (WAN)

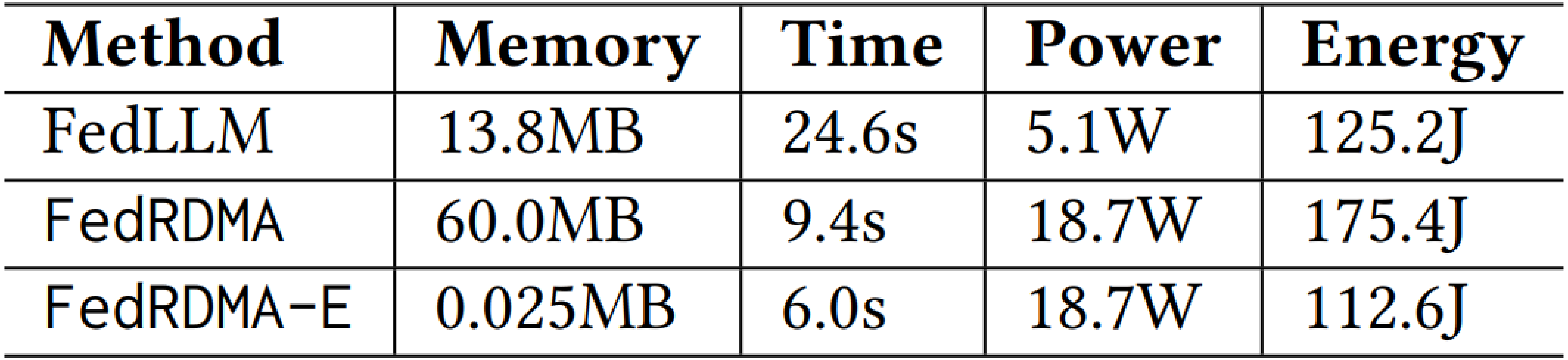
Integration with PEFT

Lora Rank	4	8	16	32	64	128	256	512	1024
Data size (MB)	1.1	2.3	4.5	9.0	18.0	36.0	72.0	144.0	288.0
Num of chunks	1	2	4	5	7	12	21	39	75
Link-Enable	NO	YES	YES	YES	YES	YES	YES	YES	YES



- FedRDMA can complement the PEFT method well.
- FedRDMA reduces communication time by over 70% in the majority of cases compared to using PEFT alone.

System Cost



Method	Memory	Time	Power	Energy
FedLLM	13.8MB	24.6s	5.1W	125.2J
FedRDMA	60.0MB	9.4s	18.7W	175.4J
FedRDMA-E	0.025MB	6.0s	18.7W	112.6J

- FedRDMA-E to achieve a 99.9% reduction in memory overhead compared to FedRDMA, demonstrating a significant improvement similar to that of FedLLM.
- FedRDMA reduces the total power consumption for transmission by more than 10%.

CONCLUSION

- **Target:** Leverage RDMA to accelerate federated learning communication on WANs.
- **Contribution:**

1. We conduct preliminary experiments to reveal high communication overhead of cross-silo FedLLM.
2. We propose FedRDMA, a communication-efficient FedLLM system featuring chunked RDMA transmission and a series of optimizations.
3. We implement FedRDMA atop FATE and conduct extensive experiments to demonstrate it saves up to 3.8× communication time compared to TCP-based FedLLM systems.

Thank you for listening!

cdq@bupt.edu.cn

Appendix

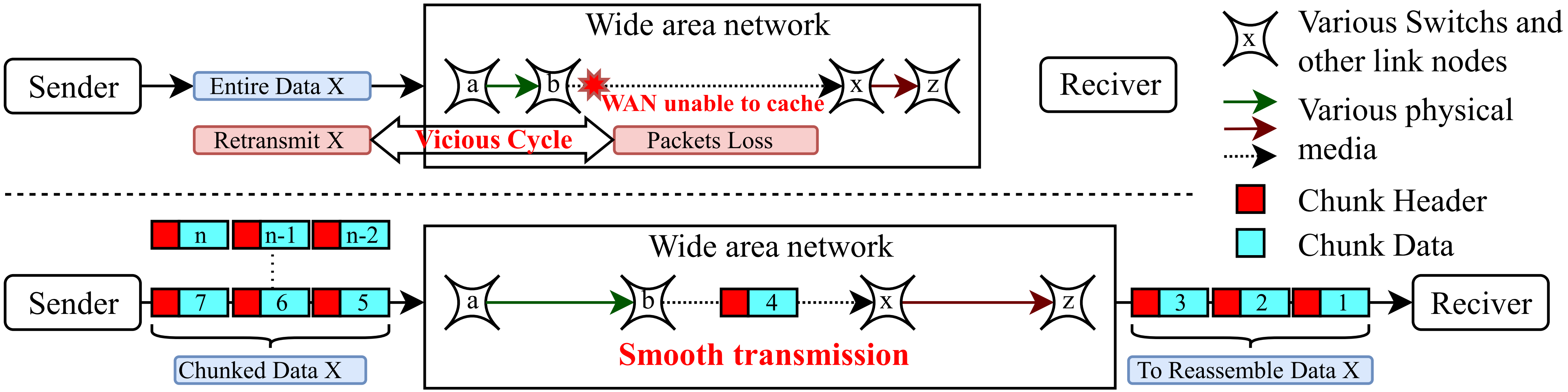
CONCLUSION AND FUTURE WORK

- **Future work:**
  - Validate FedRDMA on a wider range of models and datasets.
  - Extend FedRDMA to more complex WAN environments.
  - Facilitate FedRDMA on large-scale cross-silo federated learning deployment.

Thank you for listening!

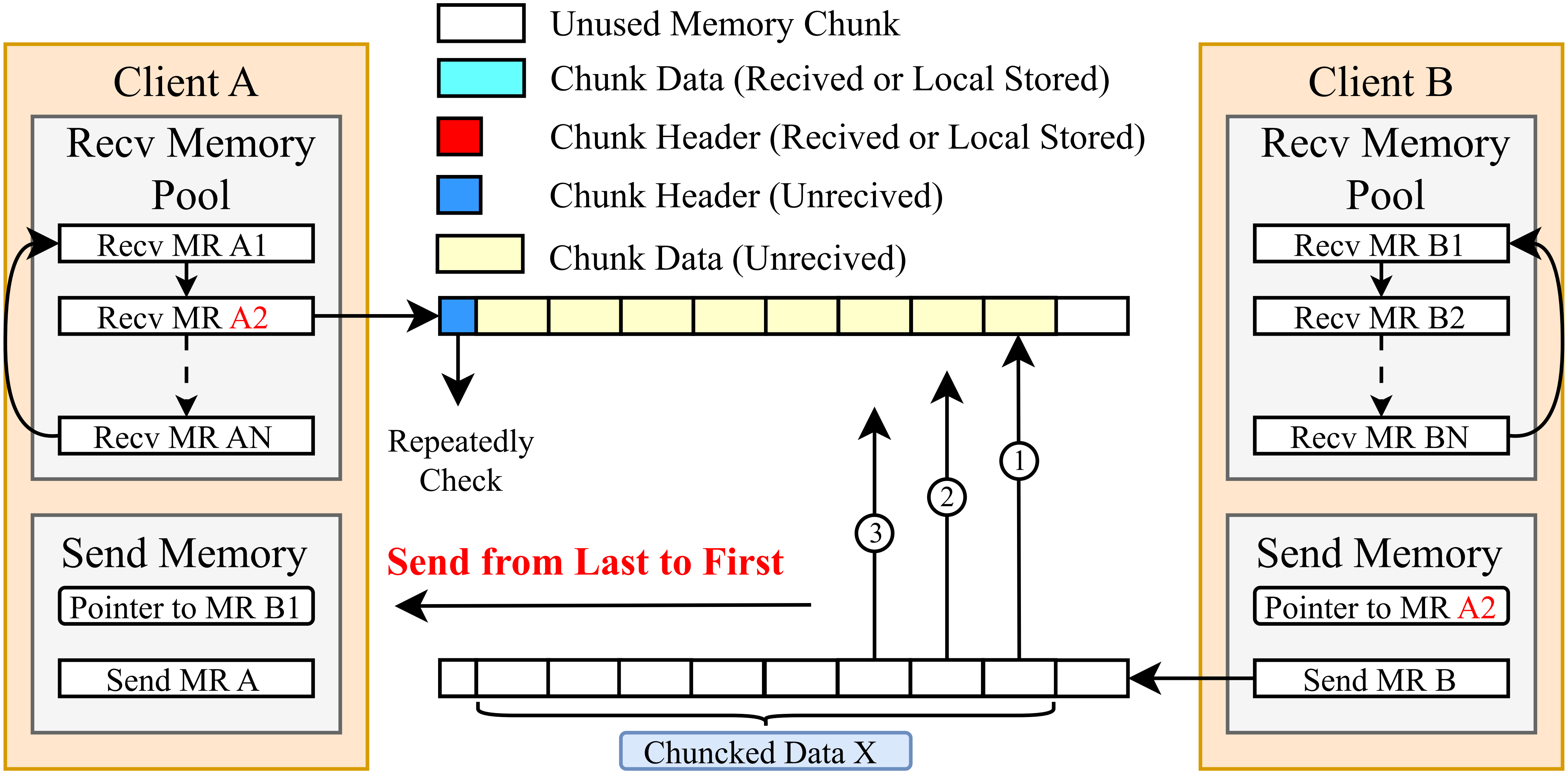
cdq@bupt.edu.cn

FedRDMA



Split X into n chunks ⇨ Send sequentially with ACKs ⇨ Parse and reassemble

Optimizations of FedRDMA



□ Unused Memory Chunk

🟦 Chunk Data (Recived or Local Stored)

🟥 Chunk Header (Recived or Local Stored)

🟦 Chunk Header (Unrecived)

🟨 Chunk Data (Unrecived)

Client A

Recv Memory Pool

Recv MR A1

Recv MR A2

Recv MR AN

Send Memory

Pointer to MR B1

Send MR A

Repeatedly Check

Send from Last to First

1

2

3

Chuncked Data X

Client B

Recv Memory Pool

Recv MR B1

Recv MR B2

Recv MR BN

Send Memory

Pointer to MR A2

Send MR B