



Towards Domain-Specific Network Transport for Distributed DNN Training

Hao Wang¹, Han Tian¹, Jingrong Chen², Xinchun Wan¹, Jiachen Xia¹,
Gaoxiong Zeng¹, Wei Bai^{3*}, Junchen Jiang⁴, Yong Wang¹, Kai Chen¹

¹iSING Lab, Hong Kong University of Science and Technology

²Duke University, ³Microsoft, ⁴University of Chicago

*Now with NVIDIA

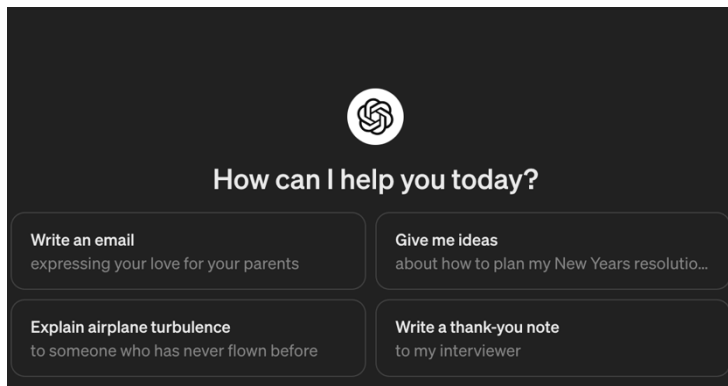
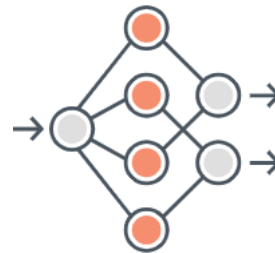
DNN empowers a wide range of applications



Face Recognition



Automatic Driving



ChatGPT



A bird scaring a scarecrow.



Paying for a quarter-sized pizza with a pizza-sized quarter.

DALL·E

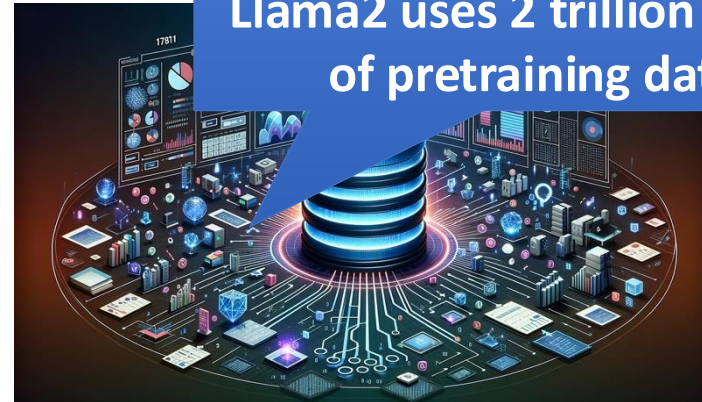
Training DNN is time-consuming

DNN model can be very complicated, with tens to hundreds of layers and millions of neurons.



Complicated models

Dataset is huge, e.g., ImageNet contains more than 14 million images. Llama2 uses 2 trillion tokens of pretraining data.



Huge amount of data

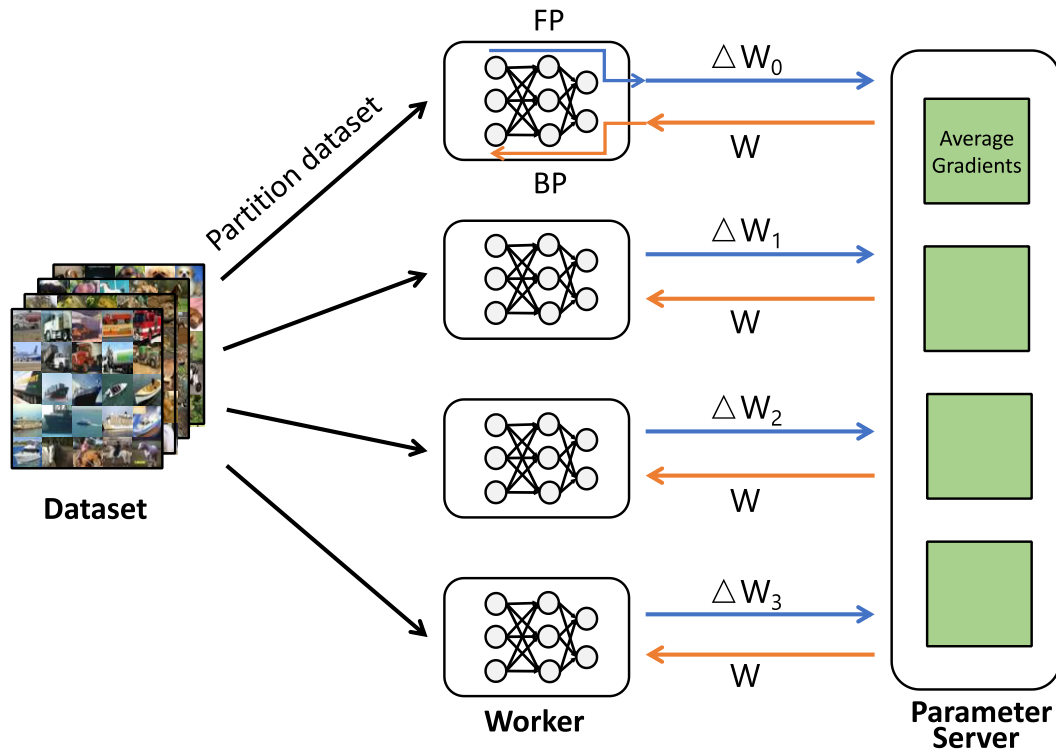
Model	BERT _{BASE}	Llama2-70B
Training time	4 days, 16 x TPU v3	1.7M GPU hours, A100

<https://arxiv.org/pdf/1810.04805.pdf>

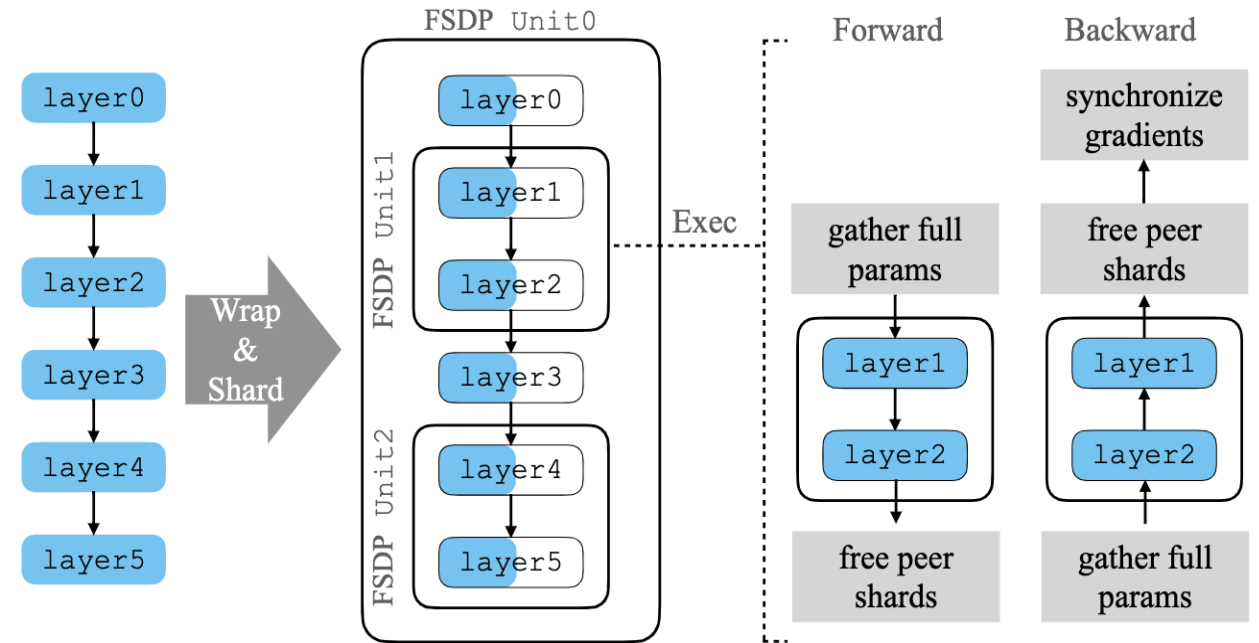
<https://arxiv.org/pdf/2307.09288.pdf>

Accelerating DNN training via data parallelism

- Example of data parallelism of synchronous SGD under the Parameter Server architecture



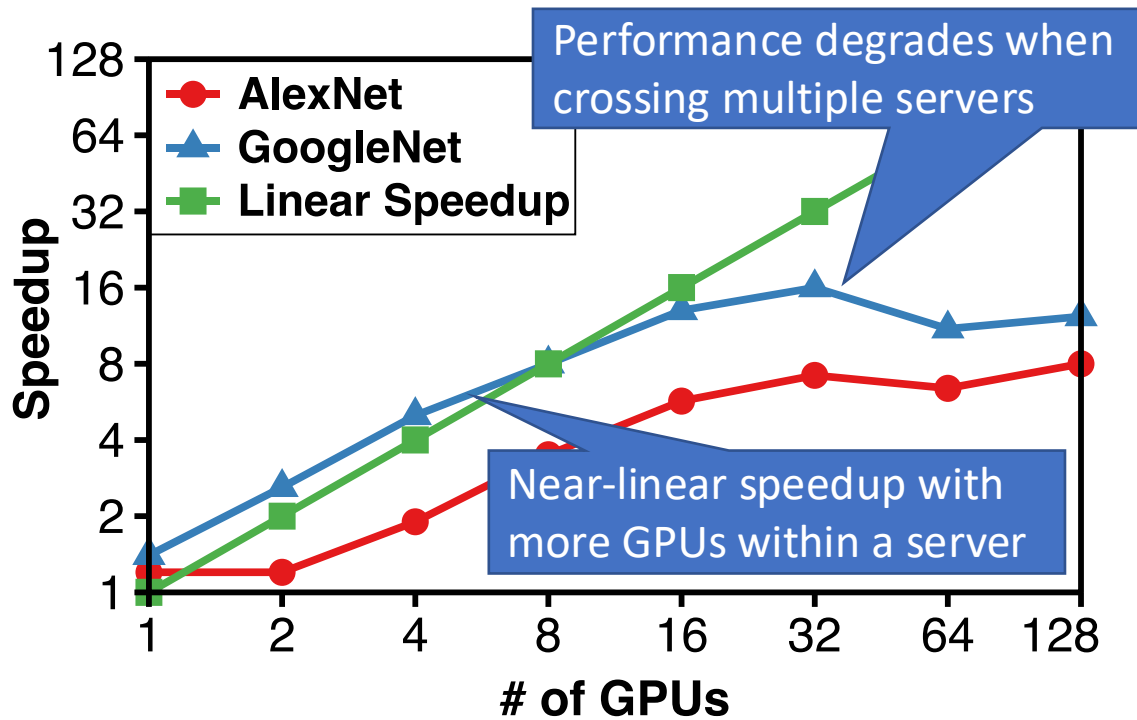
- Note that data parallelism is also widely used in LLM training, e.g., Zero and FSDP.



FSDP workflow <https://arxiv.org/pdf/2304.11277.pdf>

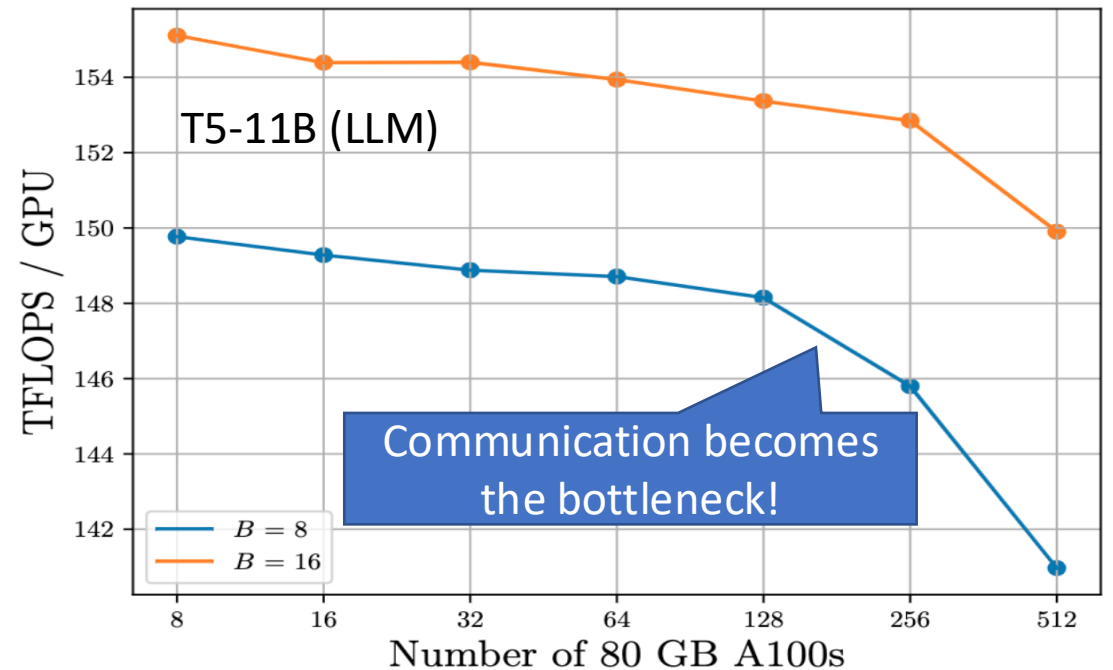
The speedup of data parallelism: a close look

➤ Speedup with more GPUs: **not always linear!**



<https://arxiv.org/pdf/1609.06870.pdf>

➤ Root cause for failing to achieve linear speedup: **communication cost!**



PyTorch FSDP: <https://arxiv.org/pdf/2304.11277.pdf>

Application layer solution: reducing traffic volume

➤ Gradient Sparsification

- Reduce communication bandwidth by only sending important gradients
- Use gradient magnitude as a simple heuristics for importance
- Only gradients larger than a threshold are transmitted (e.g., top 0.1%)

Reducing the **number** of gradients transmitted

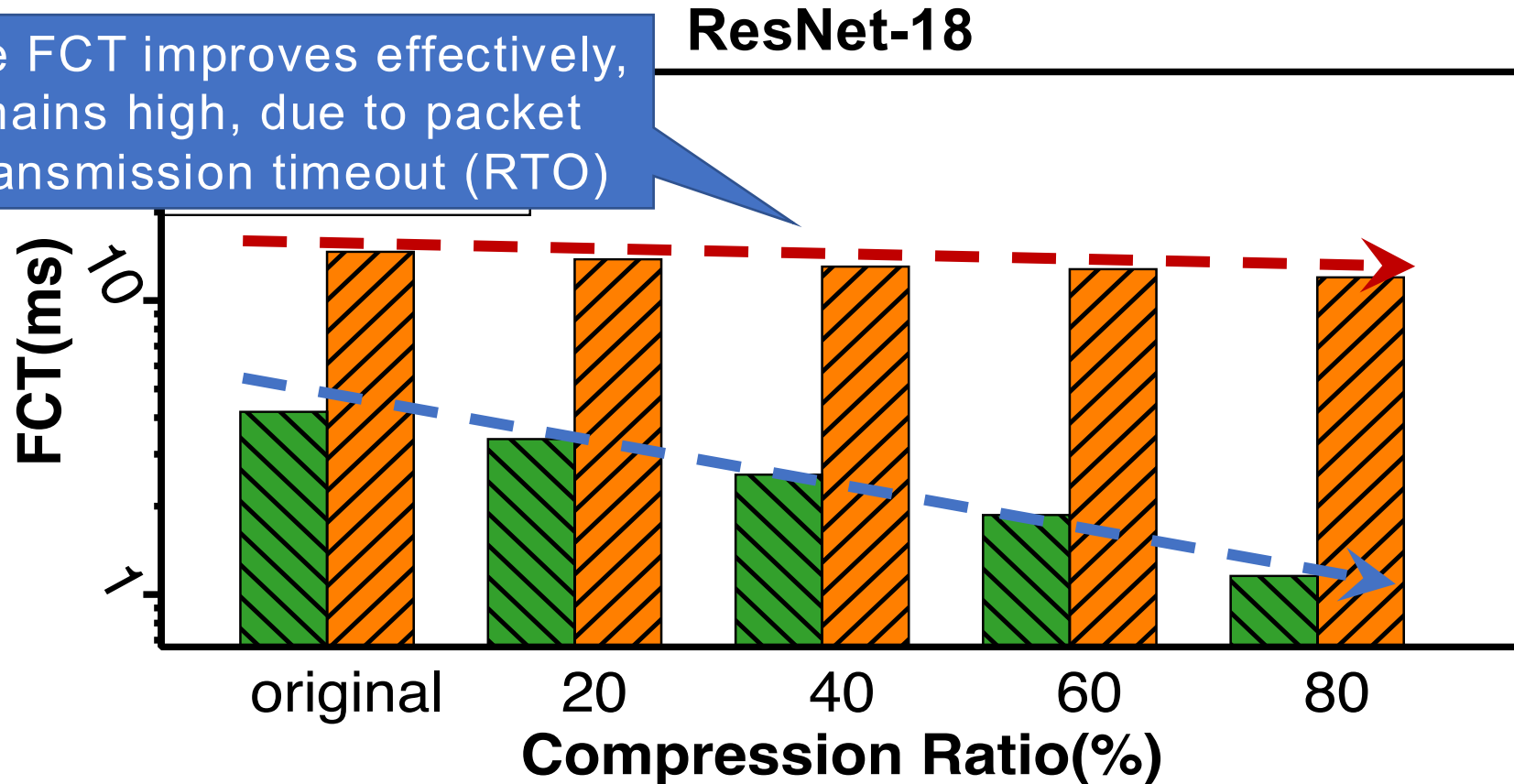
➤ Gradient Quantization

- Obtain the min and max gradient values of each layer
- Represent the gradients with low-precision float (e.g., 32 bits -> 8 bits)
- The results are composed by an array containing the quantized value, and the min and max value

Reducing the **precision** of gradients transmitted

Reducing traffic volume doesn't eliminate the problem

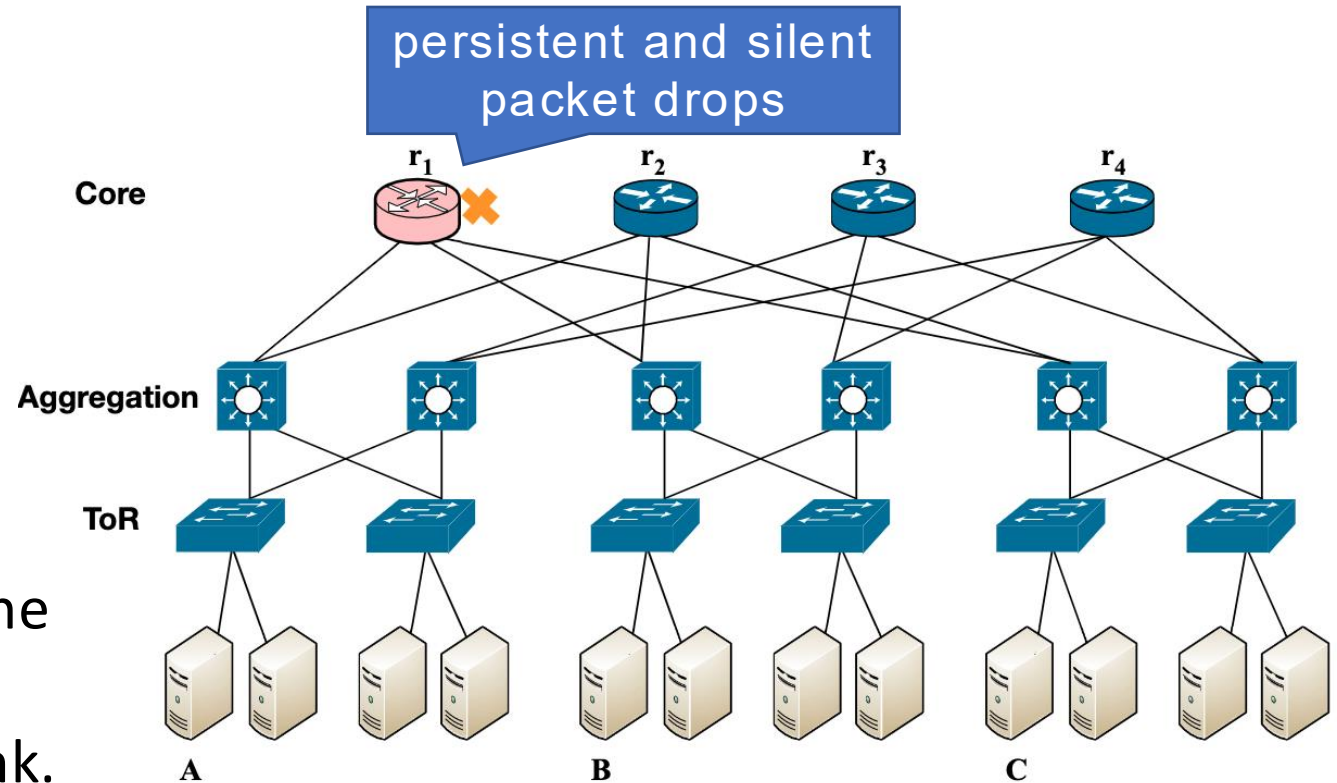
While average FCT improves effectively, tail FCT remains high, due to packet loss and retransmission timeout (RTO)



- Lesson learned: tail latency is often caused by **the communication pattern**, not only the traffic volume. **This calls for network transport solutions!**

Gray failure: potential pitfalls of large-scale training

- Fault-tolerance and reliability are crucial for distributed training
- Gray failure refers to subtle and often undetectable issues in data center
- A common example of gray failure is the persistent and silent packet drops experienced by a network device or link.

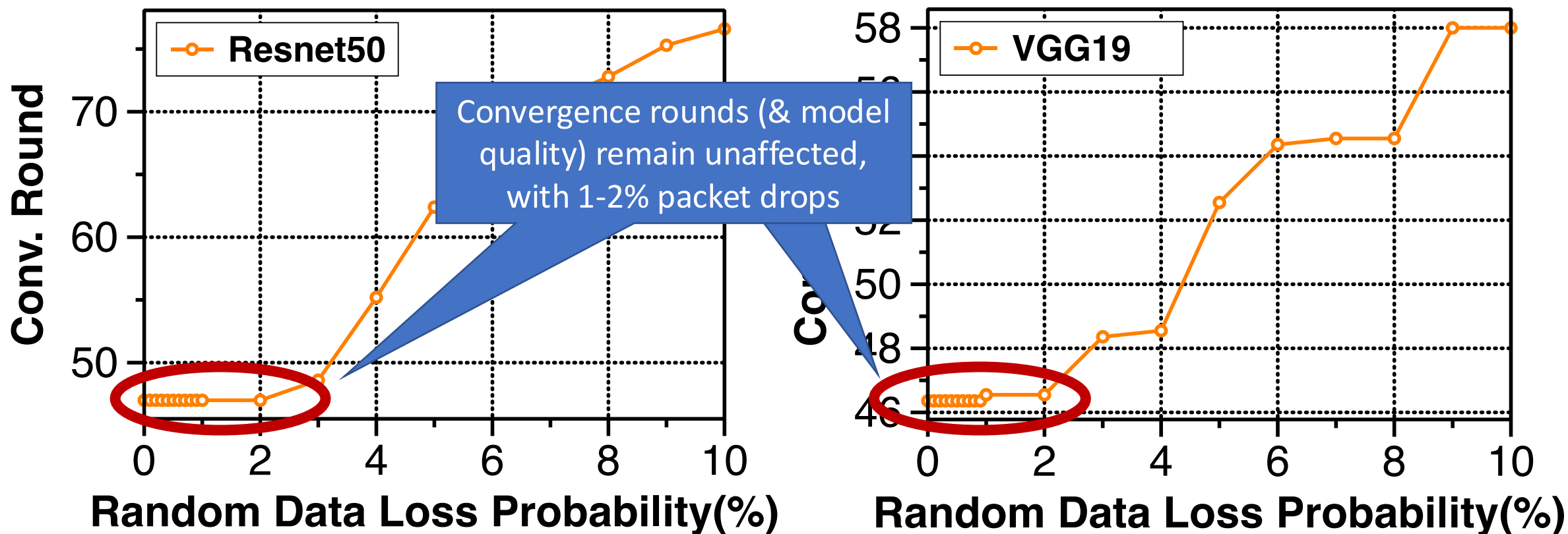


Gray Failure: The Achilles' Heel of Cloud-Scale Systems

- Transport for AI-centric Networking (AICN) must be resilient to such gray failure.

Observation 1: bounded-loss tolerance

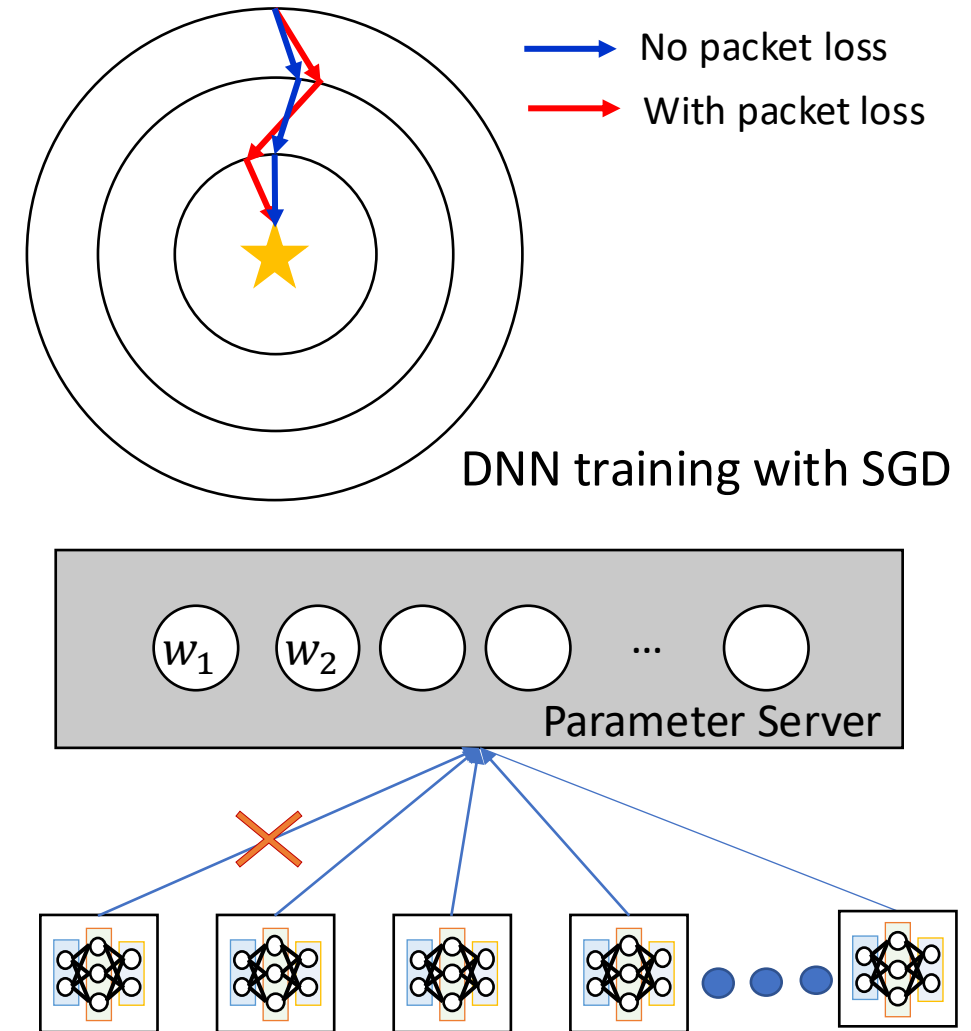
- The DNN training process is **bounded-loss tolerant**: certain packet drops don't affect model convergence much!



Dataset Used: Caltech101

Insight behind observation 1

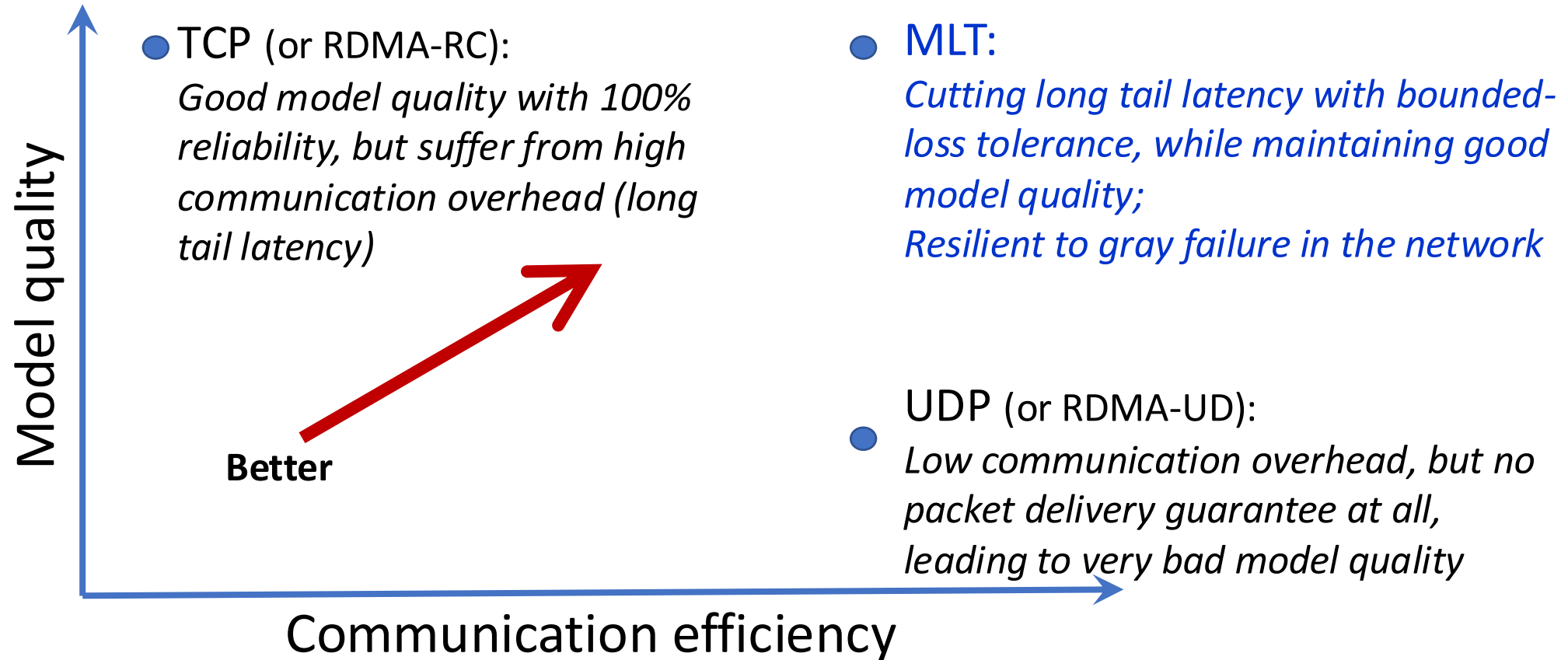
- **The learning direction doesn't deviate much:** With bounded packet losses, the direction of the gradient vector (or tensor) will not deviate much from the original, steepest direction.
- **The learning step size doesn't change much:** With bounded packet losses, the step length of the gradient vector remains similar.
- **The SGD algorithm is robust to loss (self-healing):** SGD recalculates the learning objective function towards the optimal at each step, noise caused by loss in earlier iterations won't be carried to latter iterations, but instead can be fixed later!



$$E_D[1/99 (g_2 + g_3 \dots + g_{100})] = E_D[1/100(g_1 + g_2 + g_3 \dots + g_{100})]$$

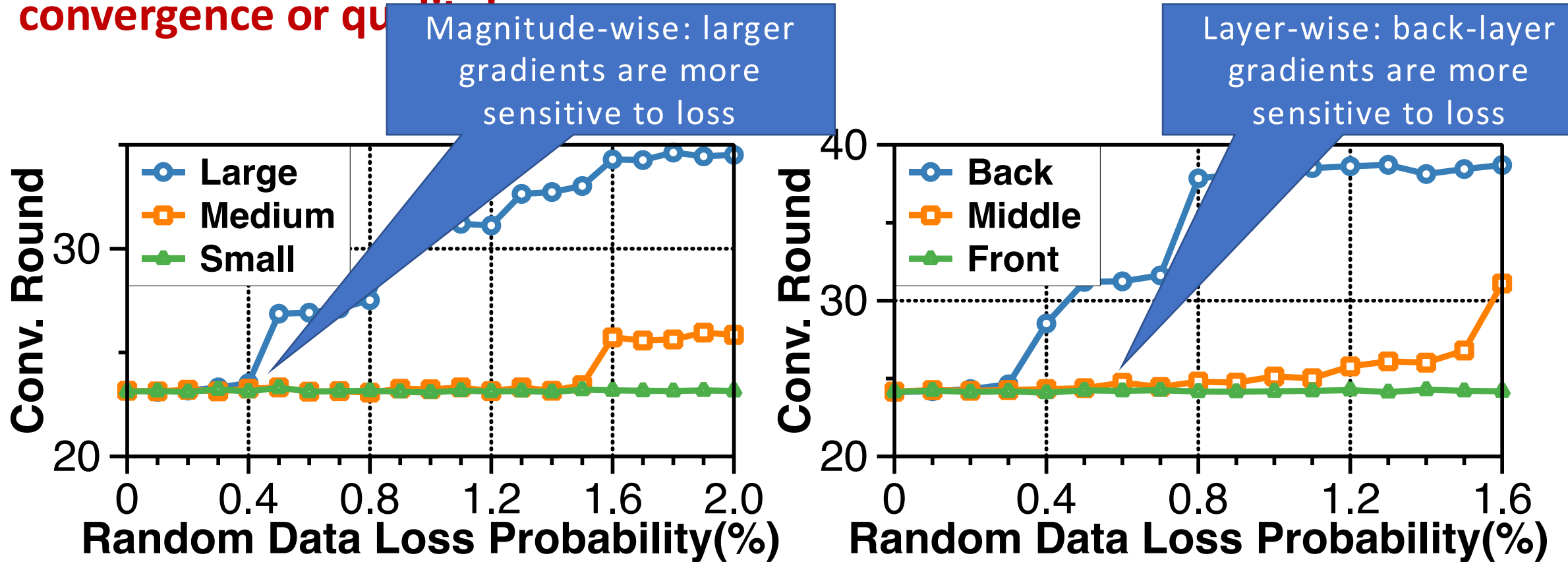
Inspiration from observation 1

➤ Reliability requirement for AI-centric Networking (AICN)



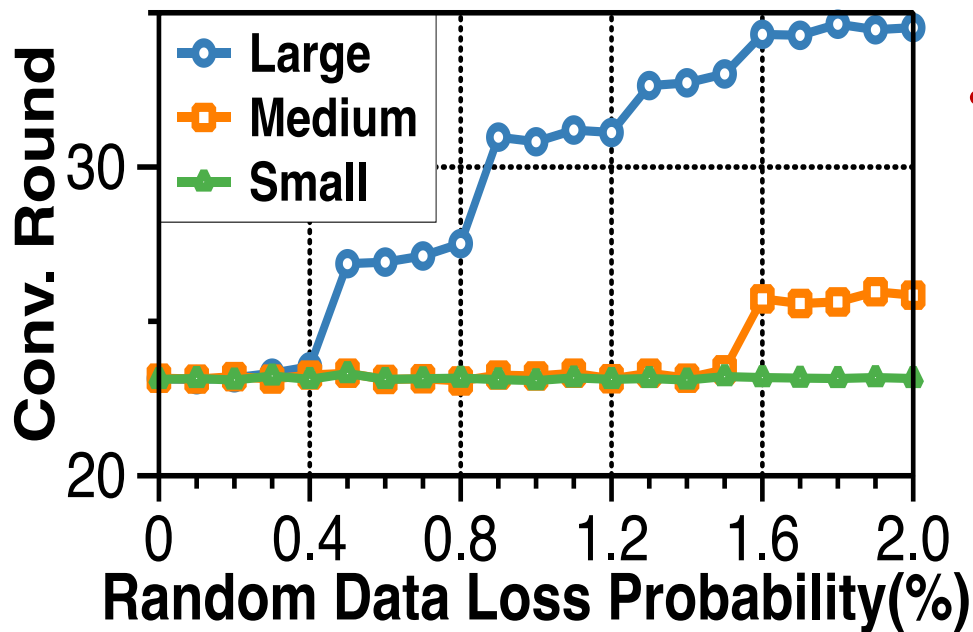
Observation 2: Different gradients have different impacts

- **Losing different gradients may generate different impacts on model convergence or quality**

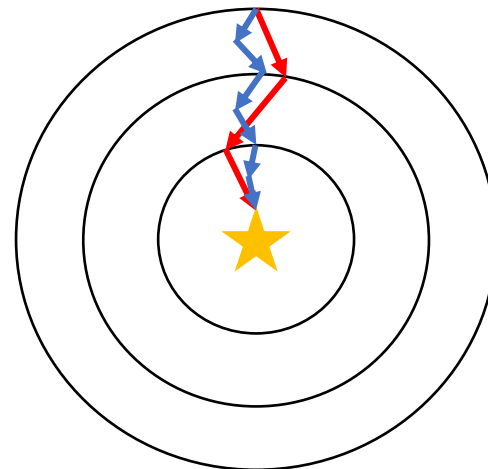


Insight behind magnitude-wise impact

➤ **Magnitude-wise impact:** larger gradients are less loss-tolerant than small gradients



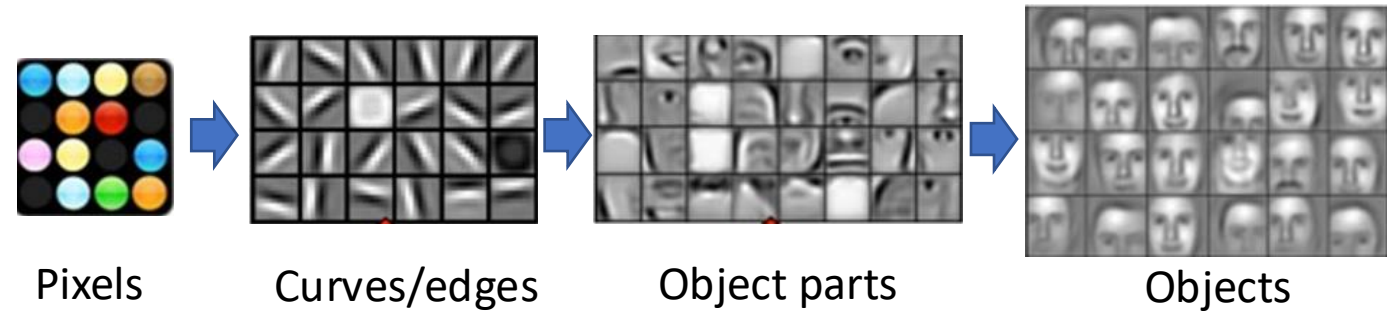
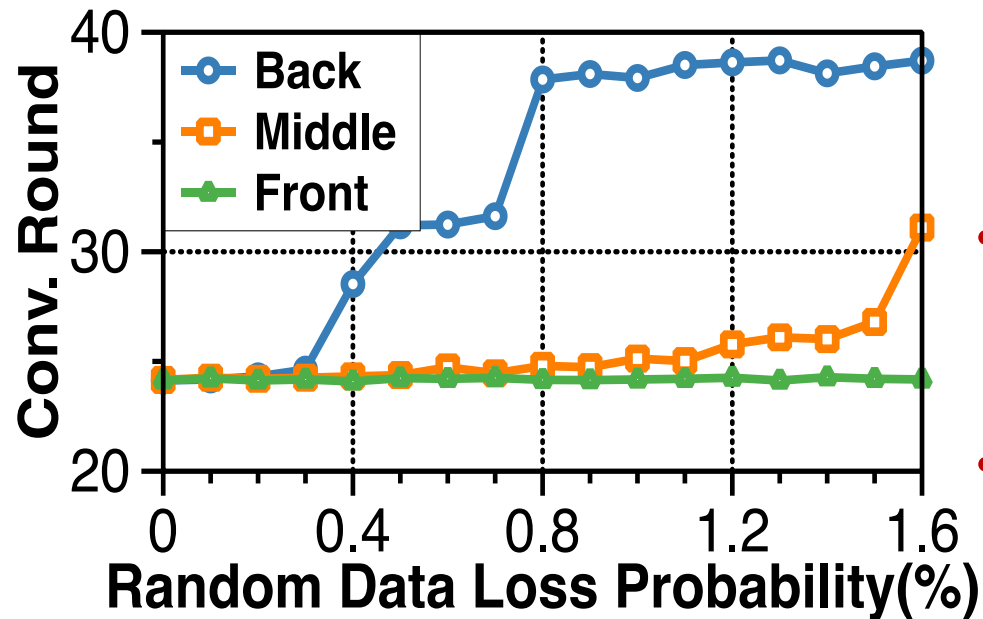
- **Larger gradient** contains stronger correlation between the extracted feature and the objective task than smaller gradient does, **more impact on model accuracy!**
- **Larger gradient** indicates bigger learning step size, smaller gradient indicates smaller step size, **more impact on convergence speed!**



- Learning step with larger gradients
- Learning step with smaller gradients

Insight behind layer-wise impact

- **Layer-wise impact:** front-layer gradients are more loss-tolerant than back-layer gradients



- **Front layers** extract simple, class-independent features and can be trained from almost all samples, e.g., from pre-training dataset, thus **easier to learn!**
- **Back layers** extract class-specific features (e.g., earrings) and can be trained only from specific samples with certain classes (e.g., women), thus **much harder to learn!**

Inspiration from observation 2

Not all gradients are equal in terms of the impacts on model convergence and training pipelining

When queue builds up

Prioritize front-layer gradients over back-layer gradients, to speed up training pipelining

Priority Queueing
(both at end-host and in network)

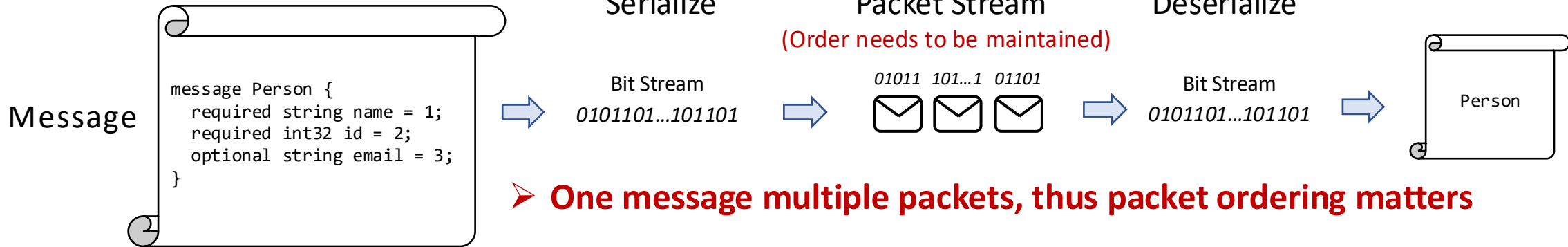
When buffer overflows

Selectively drop front-layer gradients over back-layer gradients, smaller gradients over larger gradients, to maintain model convergence/quality

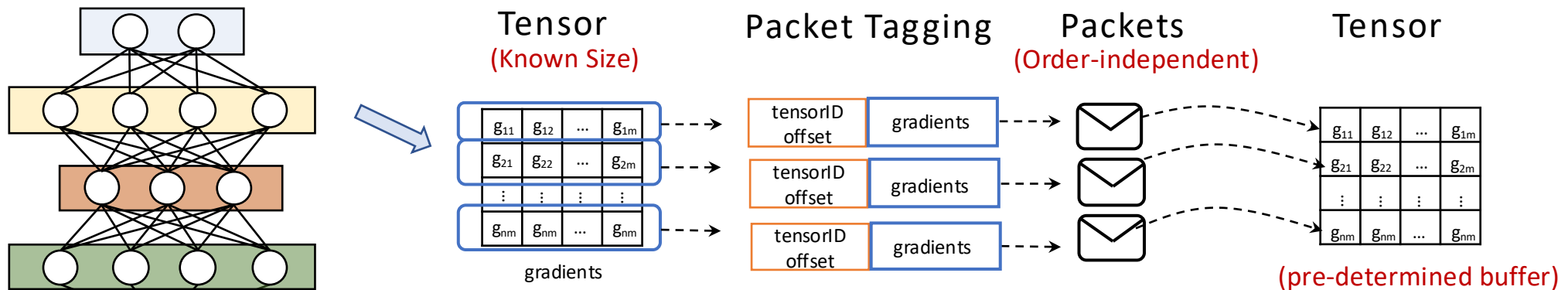
Selective Dropping

Observation 3: Inter-packet order-independence

Traditional Network Apps



DNN Training



- One packet multiple messages (gradients), thus inter-packets are order-free
- The traffic in DNN training is periodic and predictable.

Inspiration from observation 3

Tradeoff for traditional network applications

```
graph TD; A[Tradeoff for traditional network applications] --- B[Per-flow ECMP: coarse-grained, large flow hash-collision, low efficiency]; A --- C[Per-packet load balancing: fine-grained, but suffer from reordering problems]; B --> D[Flowlet-based load balancing: make a tradeoff in-between]; C --> D;
```

Per-flow ECMP:
coarse-grained, large flow
hash-collision, low efficiency

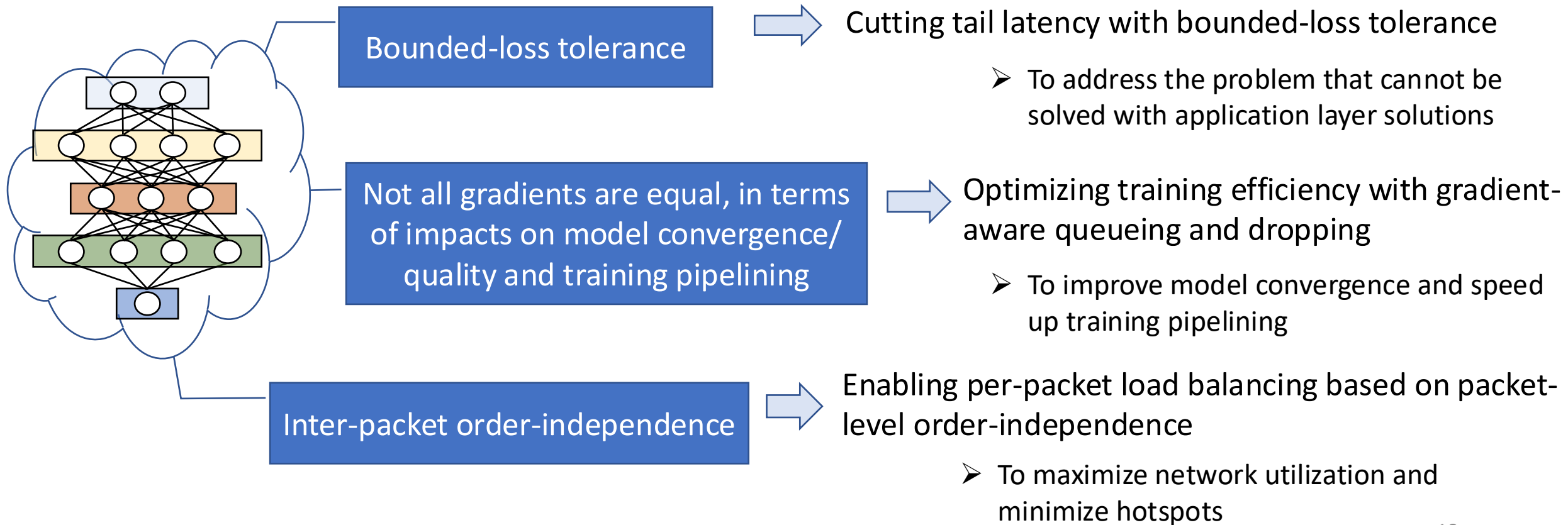
Per-packet load balancing:
fine-grained, but suffer from
reordering problems

Flowlet-based load balancing:
make a tradeoff in-between

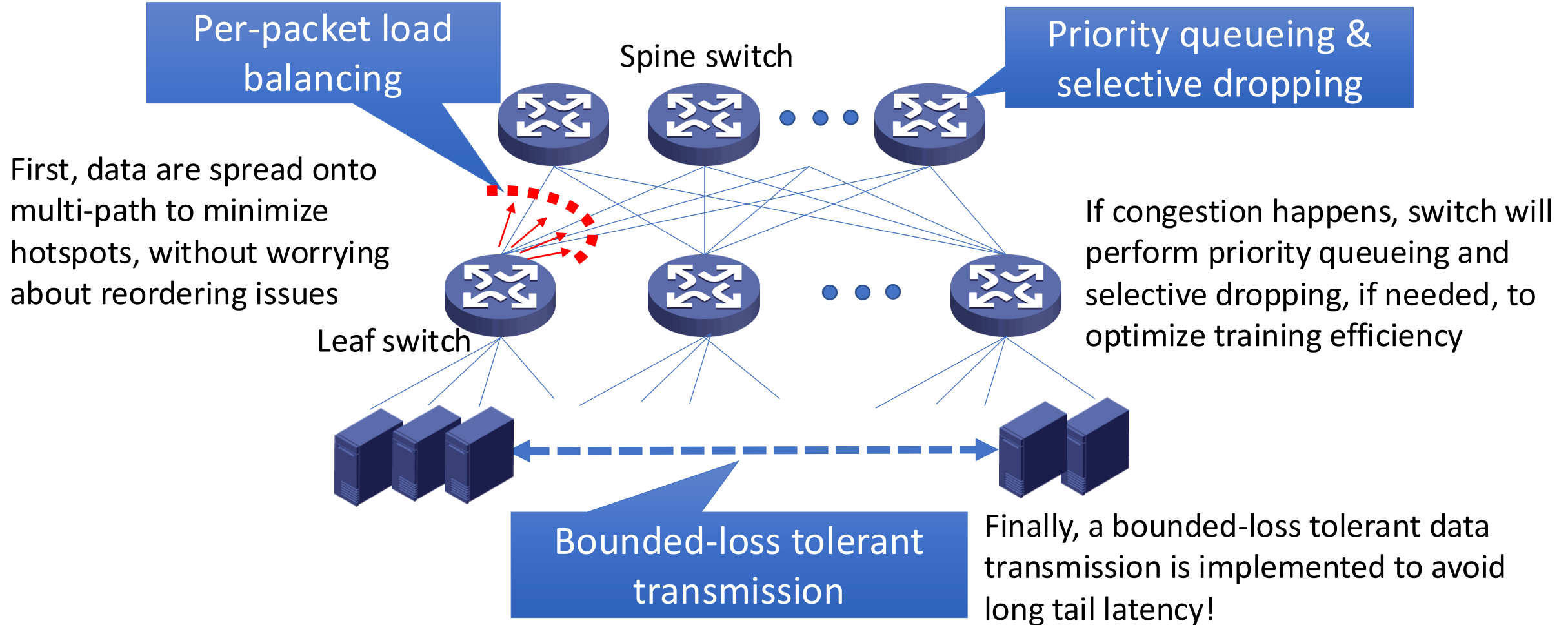
For DNN training, we can break the tradeoff: per-packet load balancing *without worrying about out-of-order issues!*

MLT - Machine Learning Transport for AI-centric networking

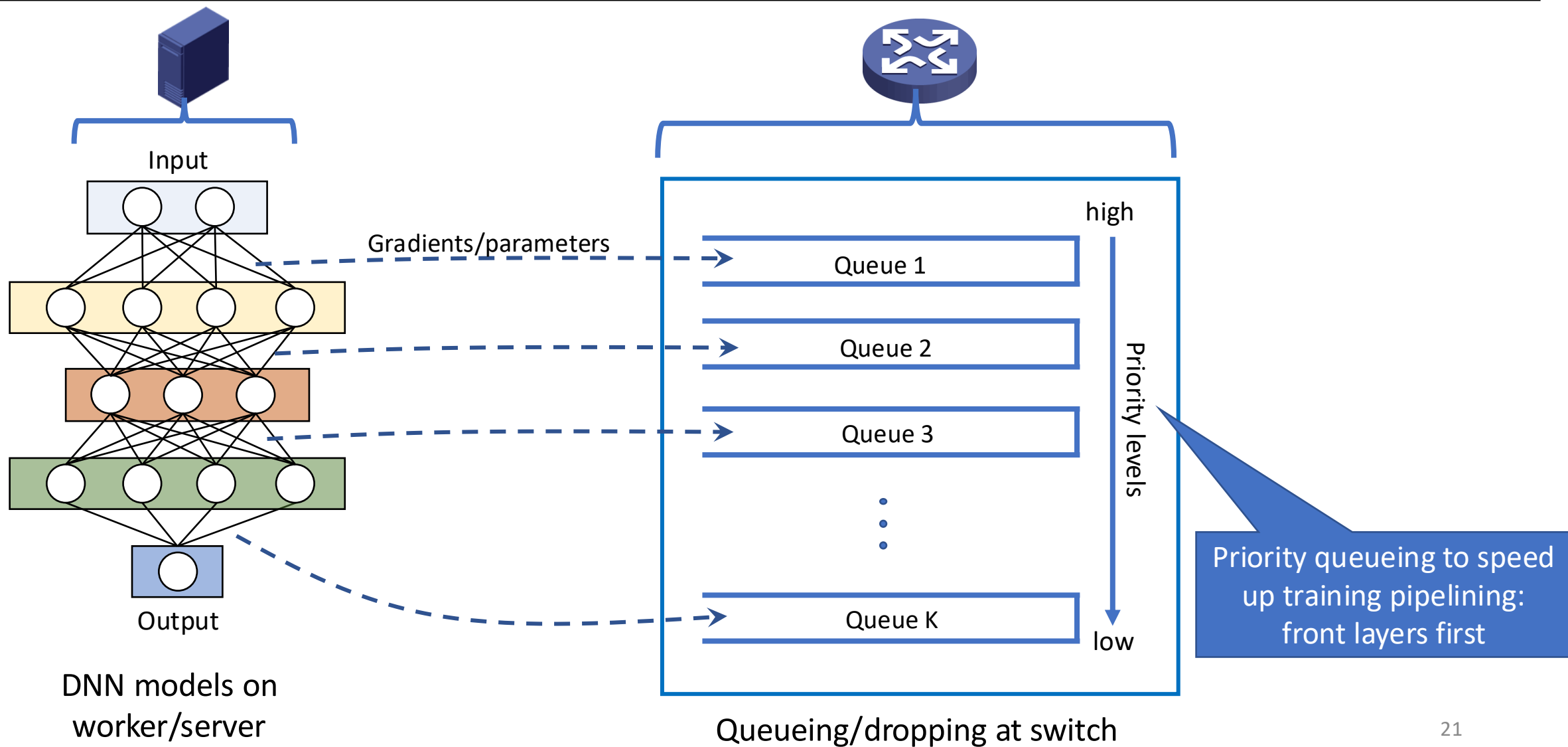
- Inspired by the previous observations, MLT performs the following domain-specific communication optimization:



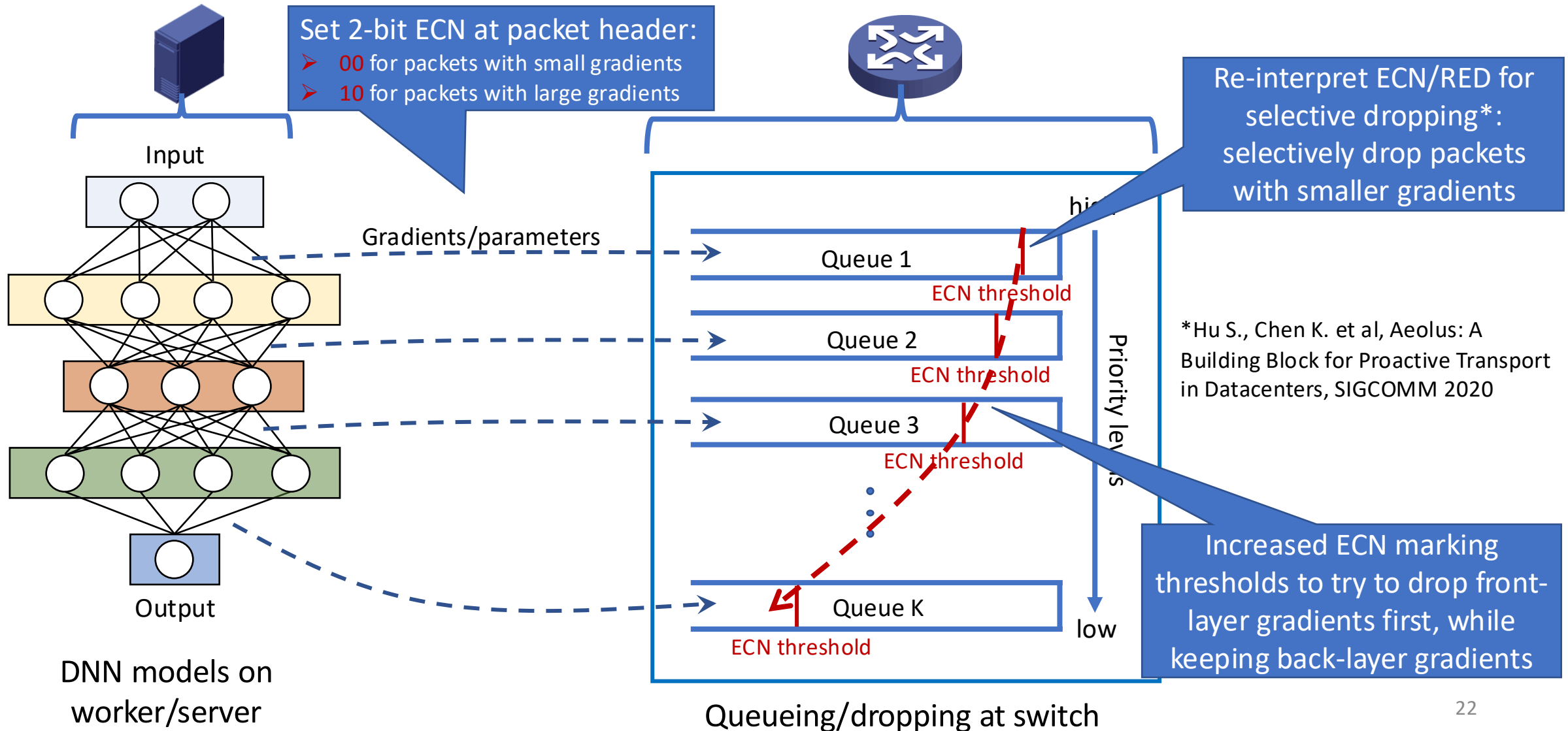
MLT design overview



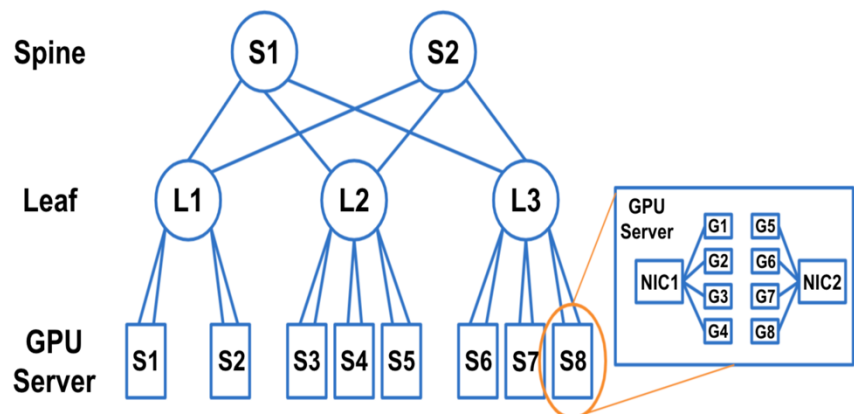
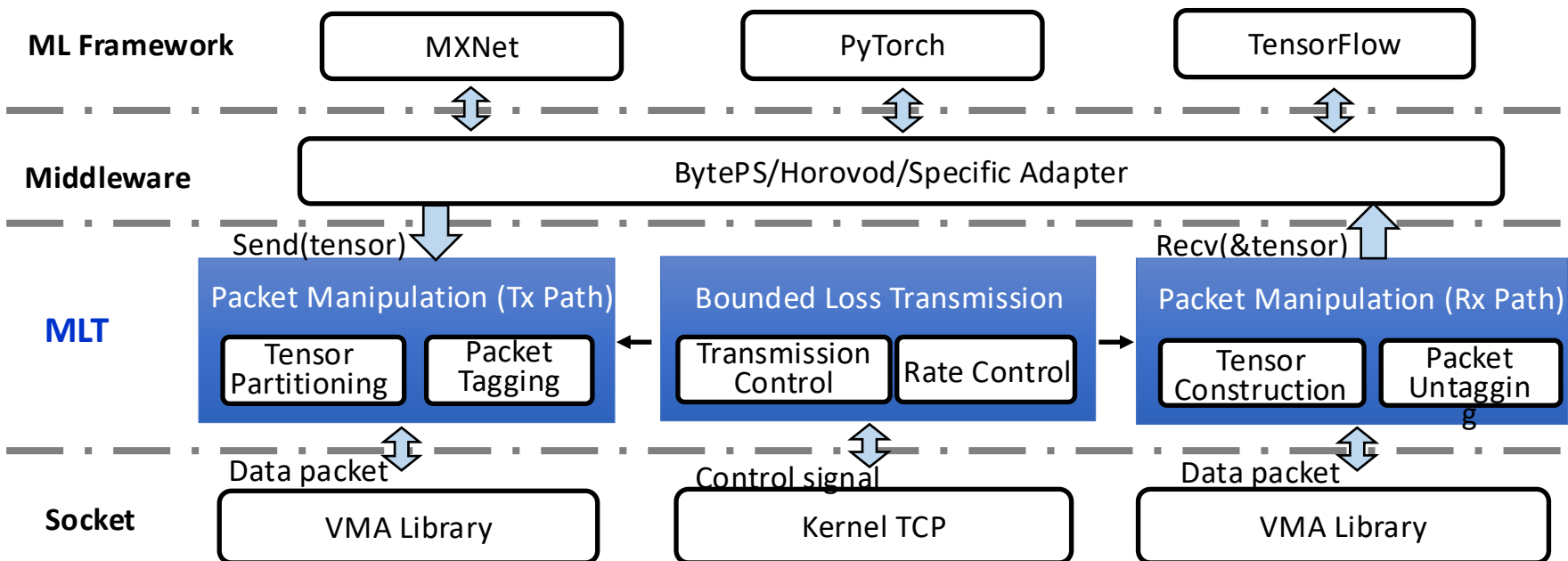
Gradient-aware **priority queueing** & selective dropping



Gradient-aware priority queueing & selective dropping



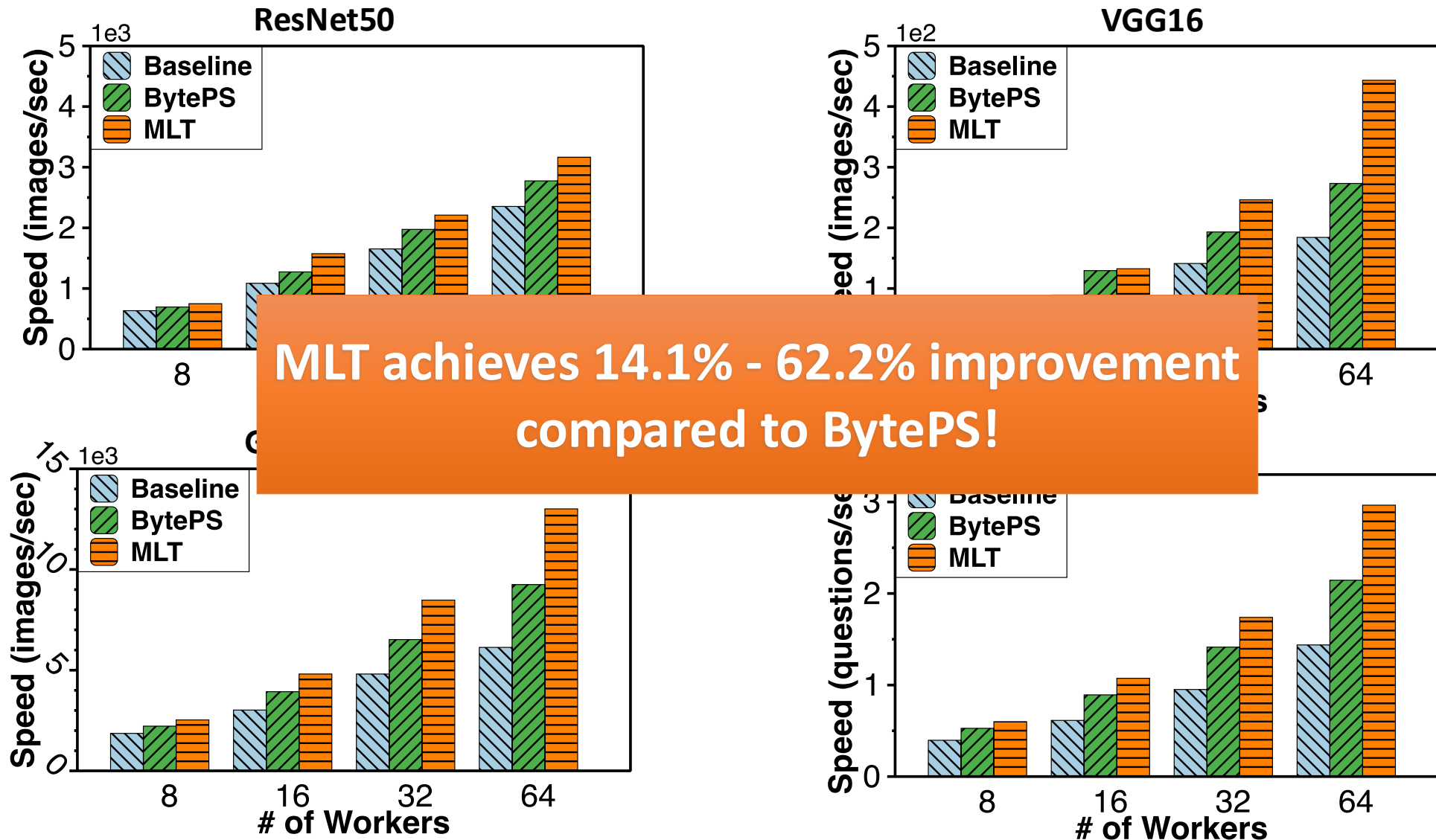
Implementation and testbed setting



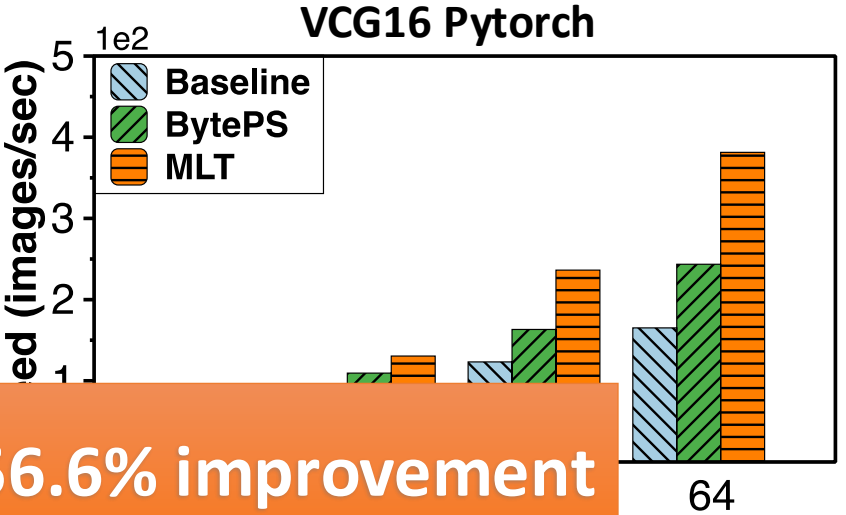
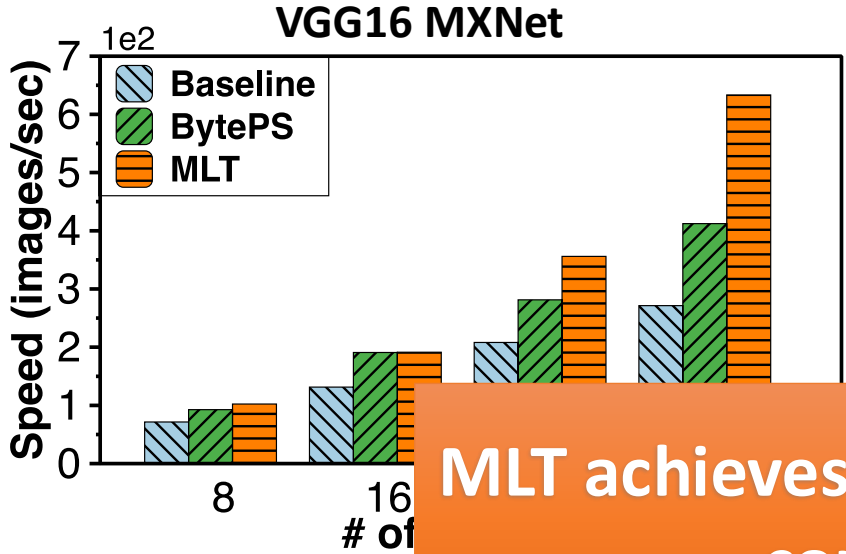
Experiment Setting:

- Testbed: 8x GPU servers each with 8x 3090 GPUs, 4 Mellanox SN2100 switches.
 - Topology: 2x3 Spine-Leaf*, 100Gbps
 - Models: ResNet50, VGG16, GoogleNet, Transformer, T5
 - Comparison Target: vanilla ML frameworks, BytePS
- *Each leaf switch has two 100Gbps links connecting to the spine switch, thus logically we have two spine switches.

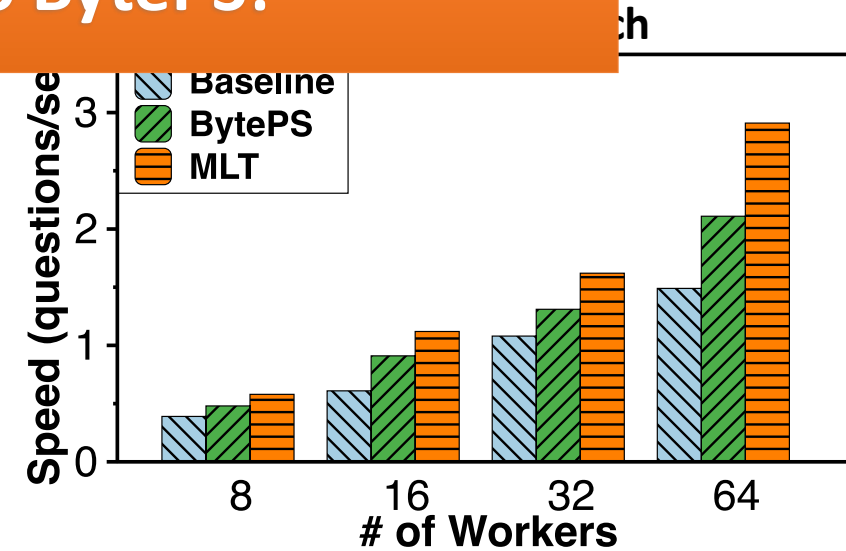
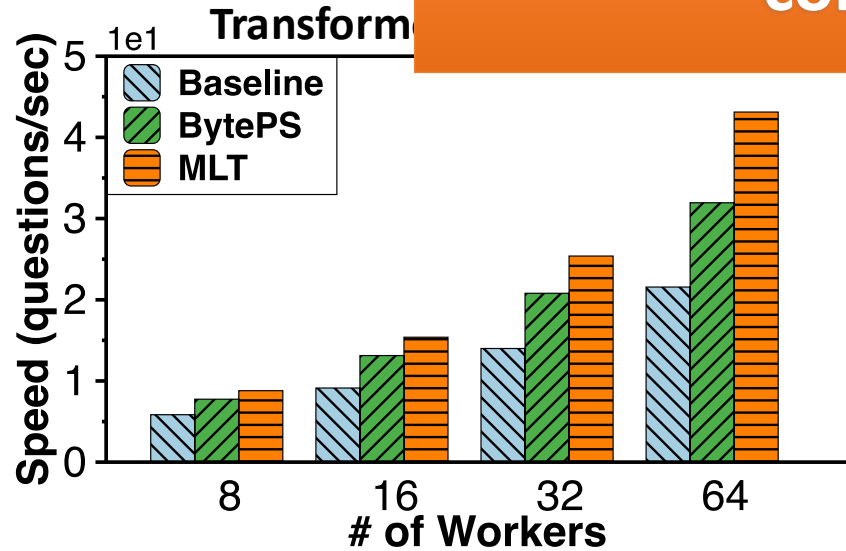
Speedup under different DNN models (Tensorflow, PS)



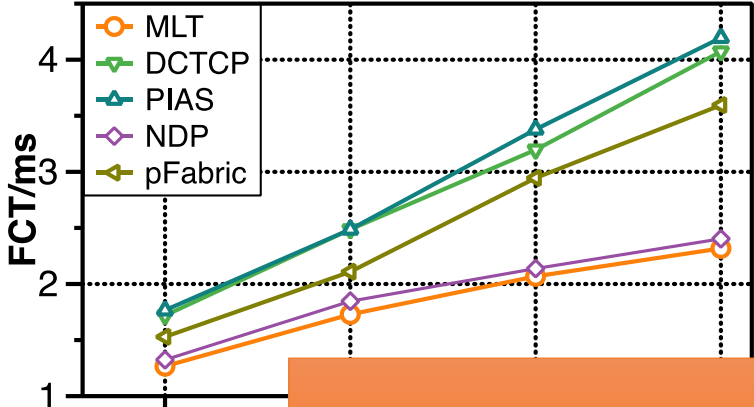
Speedup under different ML frameworks



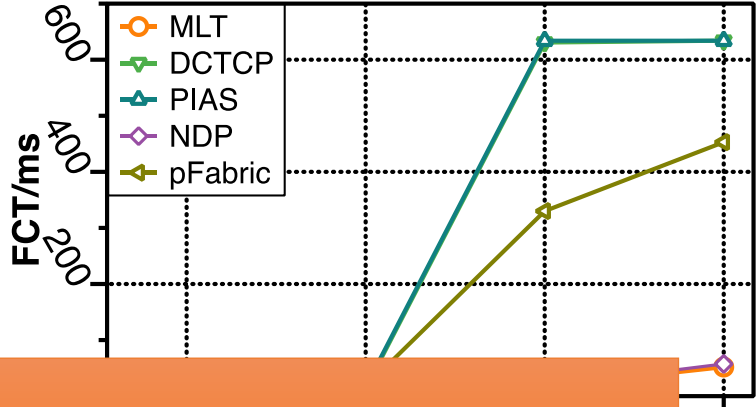
MLT achieves 12.0% - 56.6% improvement compared to BytePS!



Network performance in larger-scale simulations

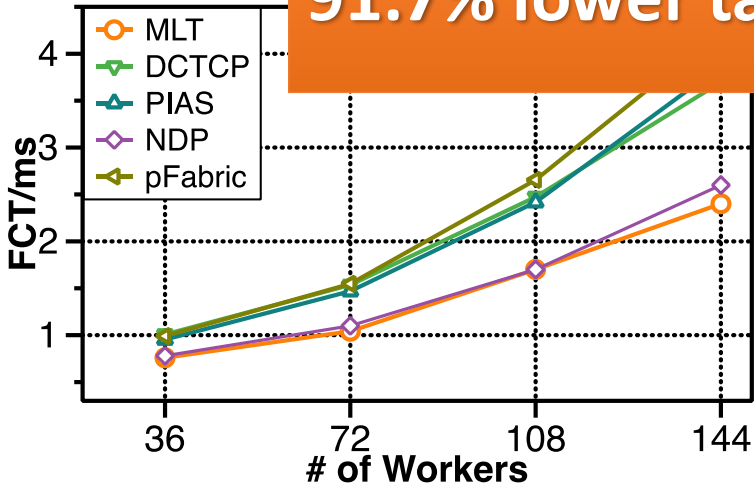


**ResNet50
Avg FCT**

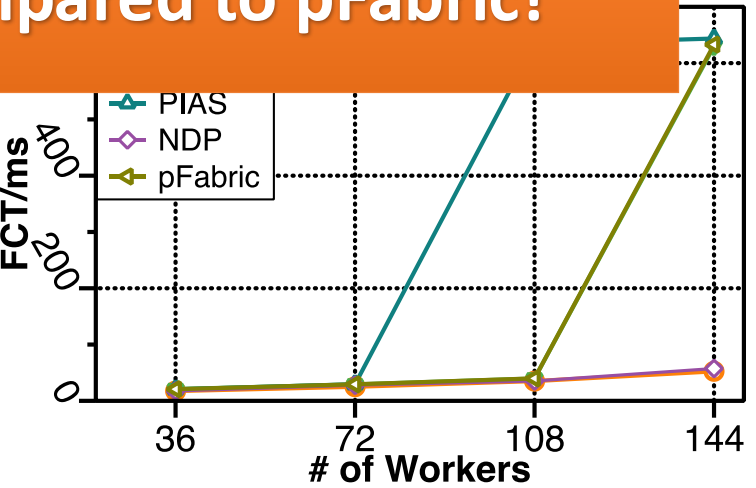


**ResNet50
Tail FCT**

MLT achieves up to 43.5% lower avg FCT and 91.7% lower tail FCT compared to pFabric!



**GoogleNet
Avg FCT**



**GoogleNet
Tail FCT**

Setting: topology 144 node leaf-spine, bandwidth 100Gbps, #servers/#workers 1/3

Conclusion

- **MLT (Machine Learning Transport for AI-centric networking)** exploits domain-specific properties of deep learning to optimize communication for distributed DNN training!
- **MLT made three key observations:**
 - Bounded-loss tolerance
 - Different gradients generate different impacts
 - Inter-packet order-independence
- **MLT conceived three main ideas:**
 - Cutting tail latency via bounded-loss tolerant data transmission
 - Improving training efficiency through gradient-aware priority queueing and selective dropping
 - Maximizing network utilization by enabling per-packet load balancing due on inter-packet order-independence

Thank you!