


RoundRole: Unlocking the Efficiency of Multi-party Computation with Bandwidth-aware Execution

Xiaoyu Fan, Kun Chen, Jiping Yu, Xin Liu, Yunyi Chen, and Wei Xu

NDSS 2026

Thanks Wenjun Yang for presentation!

Secure Multi-party Computation.

- Secure Multi-party Computation (MPC) enables multiple parties to jointly compute a function $\mathcal{F}$ on their private inputs to obtain the correct results while protecting the inputs data.
- One commonly-used scheme, Secret Sharing (SS) splits inputs to multiple random shares, and implements $\mathcal{F}$ by designing corresponding protocols.
- Joint parties communicating with each other following the protocol.

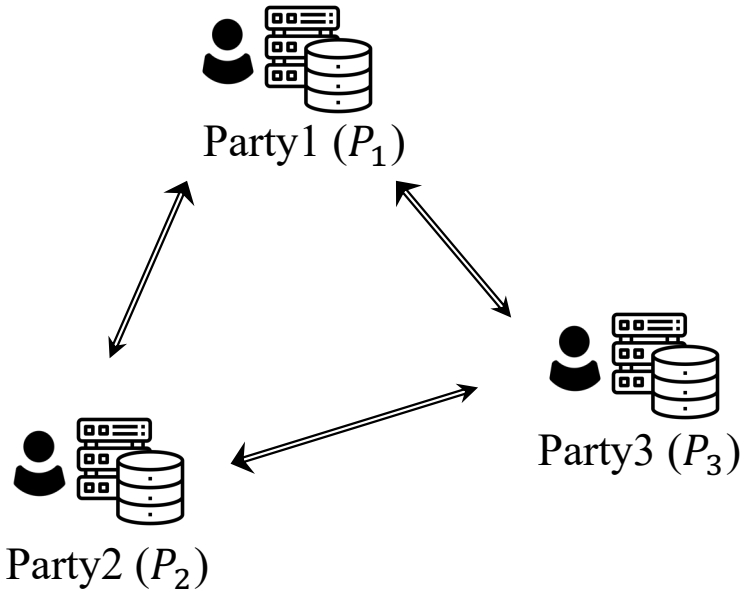


Diagram of MPC

The performance of MPC is constrained by the communications.

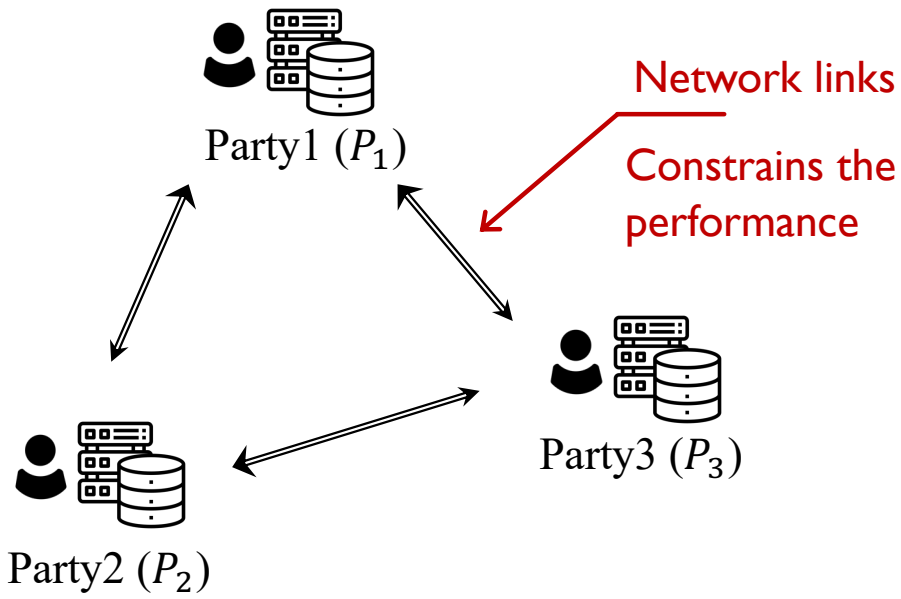


Diagram of MPC

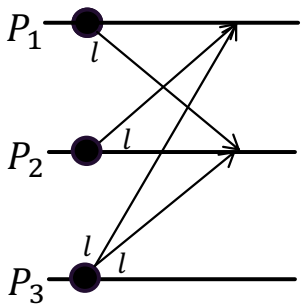
- During the execution of MPC protocols, the performance is constrained by the cross-party communications.
- Classic bandwidths:
  - LAN - 10 Gbps
  - WAN - 100 Mbps ~ 1Gbps
- However, modern CPU can process approx 30 ~100 GB/s.
- Therefore, cross-party communication often makes the CPU waiting for data.

Communication optimization often leads to asymmetric communication patterns.

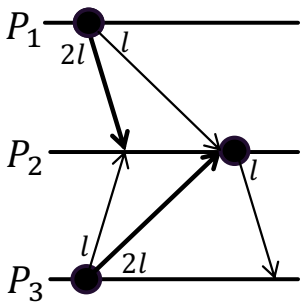
- Then, the community has done amazing jobs reducing the amount of the data to send and the number of communication rounds to improve the communication efficiency.

Communication optimization often leads to asymmetric communication patterns.

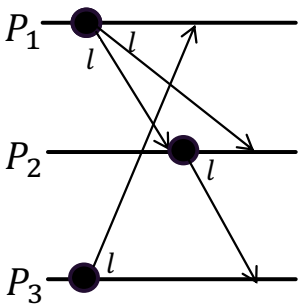
- Then, the community has done amazing jobs reducing the amount of the data to send and the number of communication rounds to improve the communication efficiency.
- During the optimization, we found that many state-of-the-art protocols include asymmetric communication patterns.



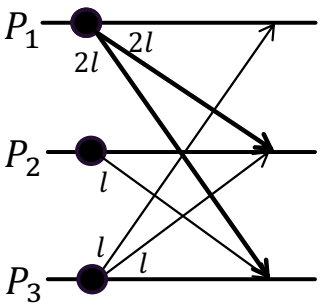
Mul (Fixed-point)



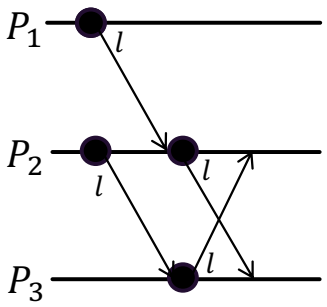
Arith to Bool



Bool to Arith



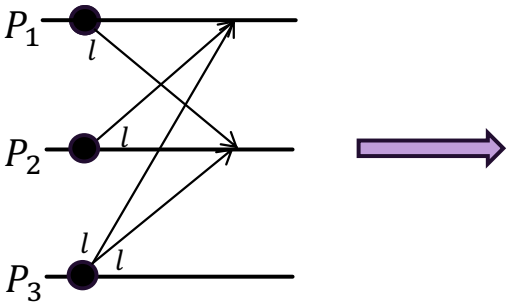
Shuffle



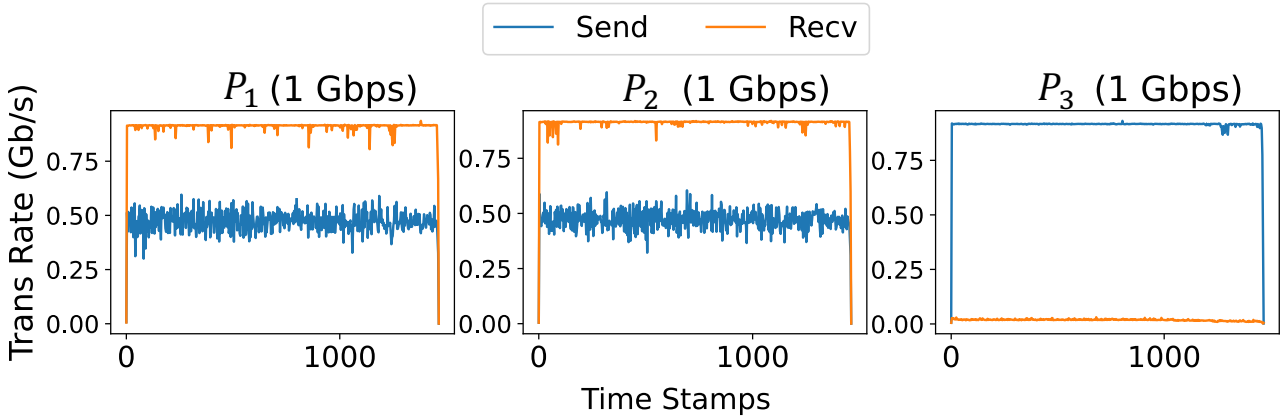
Trio Mul (Online)

- Sending and receiving volume varies for different parties.

However, the asymmetry "locks" the efficiency.



Mul (Fixed-point)



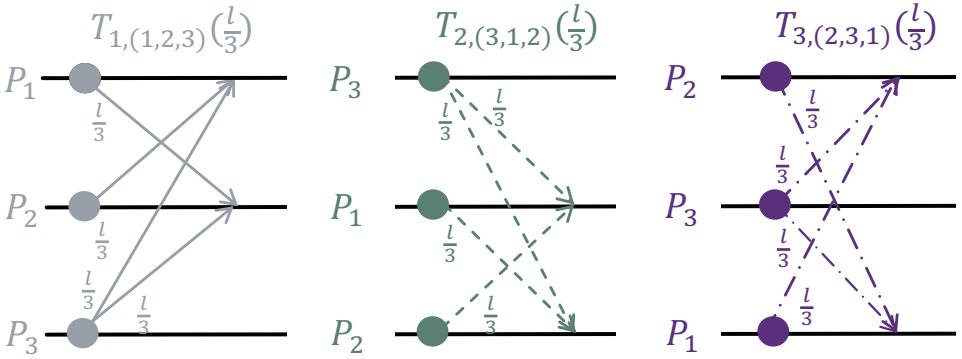
- The receiving capacity of P_3 is totally wasted – no receiving bits.
- The sending capacities of P_1 and P_2 are 50% wasted – constrained by the receiving capacity of P_2 and P_1.
- **33.3% bandwidths are wasted in total!**

Goal and Idea: "Unlocking" the efficiency by decoupling logical roles from the physical nodes.

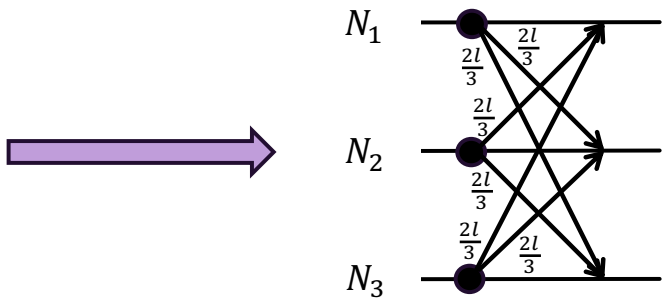
- By decoupling the logical party role (P_1 , P_2 and P_3) from the physical nodes (N_1 , N_2 and N_3),
- We can split the protocol into multiple parallel tasks and assign the role (determining the communication requirement by the protocol) of each task to the physical nodes (determining the physical bandwidth) to make full use of the bandwidth.

Goal and Idea: "Unlocking" the efficiency by decoupling logical roles from the physical nodes.

- By decoupling the logical party role (P_1 , P_2 and P_3) from the physical nodes (N_1 , N_2 and N_3),
- We can split the protocol into multiple parallel tasks and assign the role (determining the communication requirement by the protocol) of each task to the physical nodes (determining the physical bandwidth) to make full use of the bandwidth.



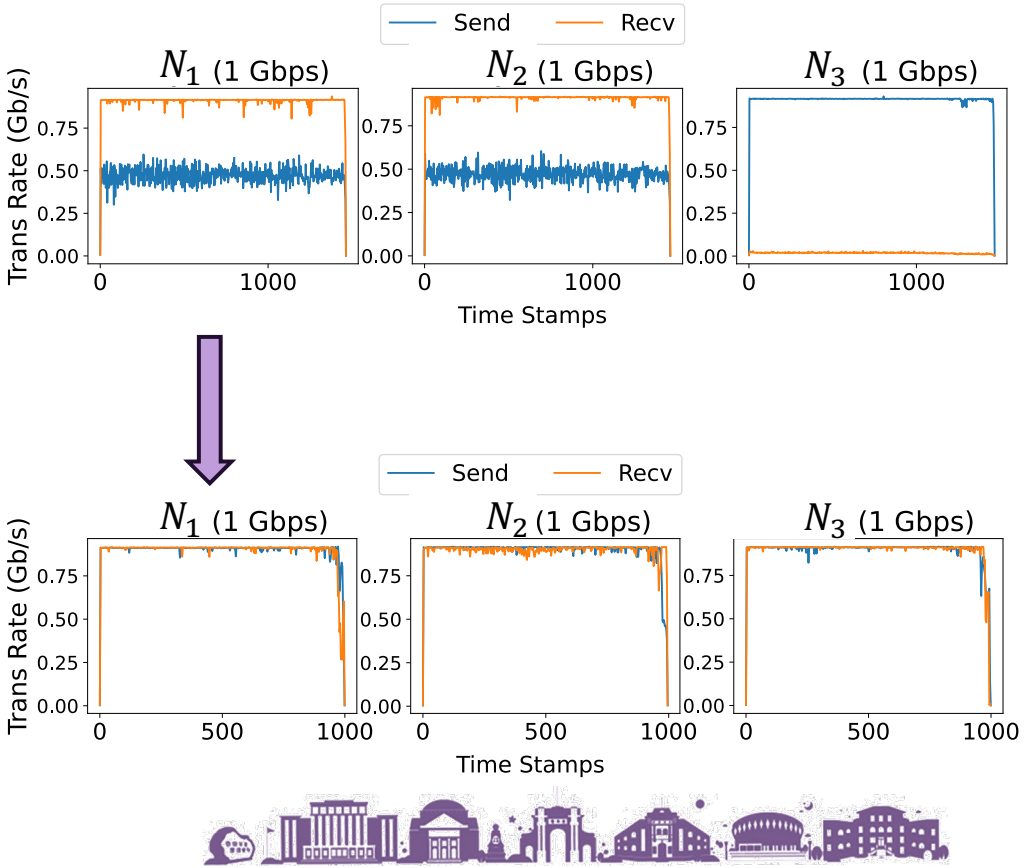
Communication Diagram - of Each Task.



- Overall.

Goal and Idea: "Unlocking" the efficiency by decoupling logical roles from the physical nodes.

- By decoupling the logical party role (P_1 , P_2 and P_3) from the physical nodes (N_1 , N_2 and N_3),
- We can split the protocol into multiple parallel tasks and assign the role (determining the communication requirement by the protocol) of each task to the physical nodes (determining the physical bandwidth) to make full use of the bandwidth.
- Thereby improving the bandwidth utilization to 100%!



RoundRole: Automatic execution-level optimization.

RoundRole: Automatic execution-level optimization.

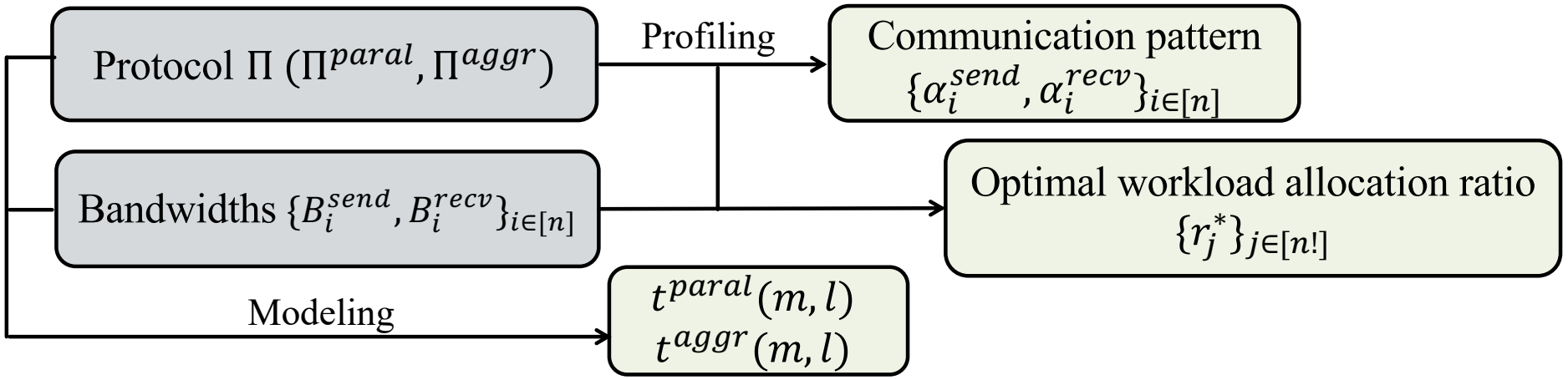
- Based on the above motivation, we design RoundRole, which begins with the protocol and the physical bandwidths.

Protocol Π $(\Pi^{paral}, \Pi^{aggr})$

Bandwidths $\{B_i^{send}, B_i^{recv}\}_{i\in[n]}$

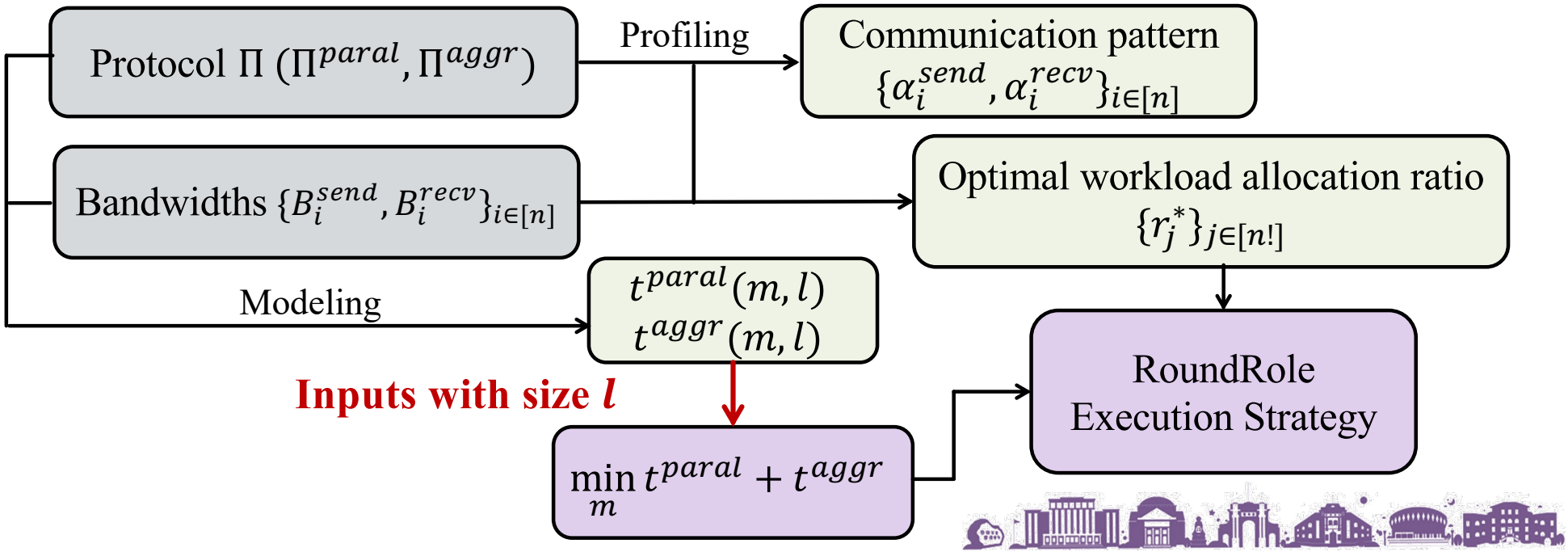
RoundRole: Automatic execution-level optimization.

- Then, RoundRole automatically profiles three sets of **input-size-independent** parameters from the protocol and the bandwidths capturing the performance characters.



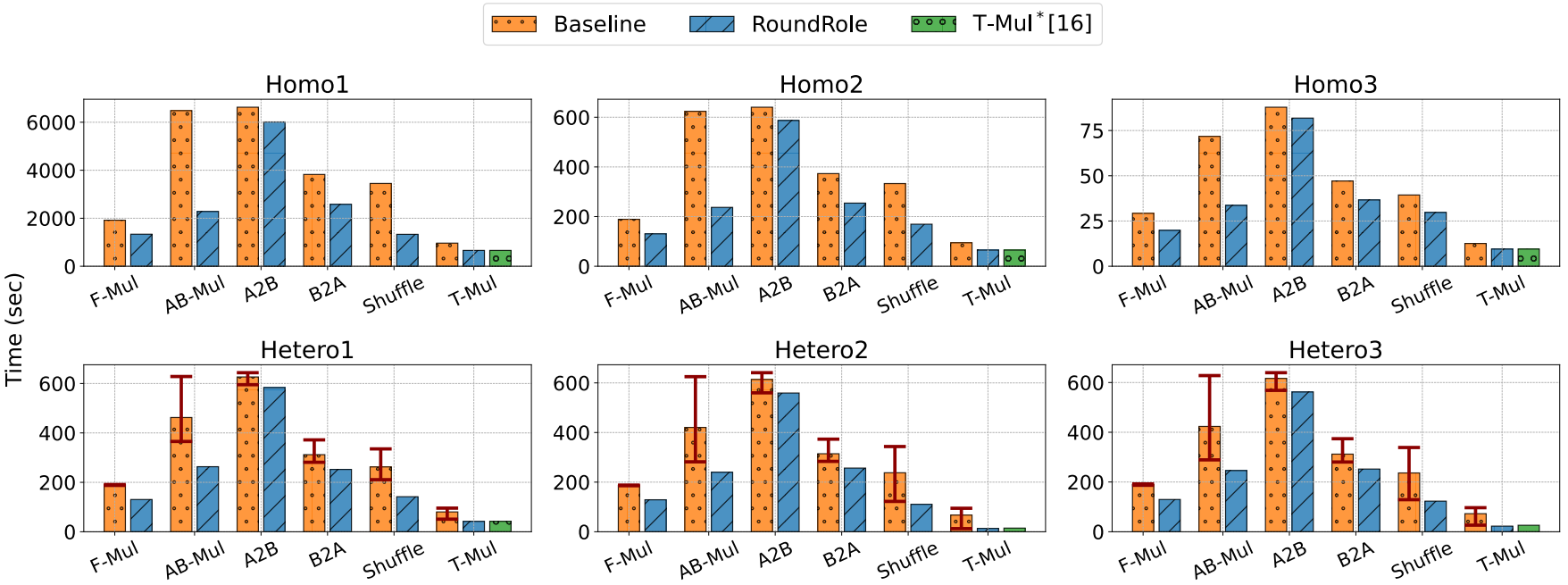
RoundRole: Automatic execution-level optimization.

- When given the inputs with size l, RoundRole then decides the execution strategy promptly – including the number of tasks and the corresponding mappings of each task.

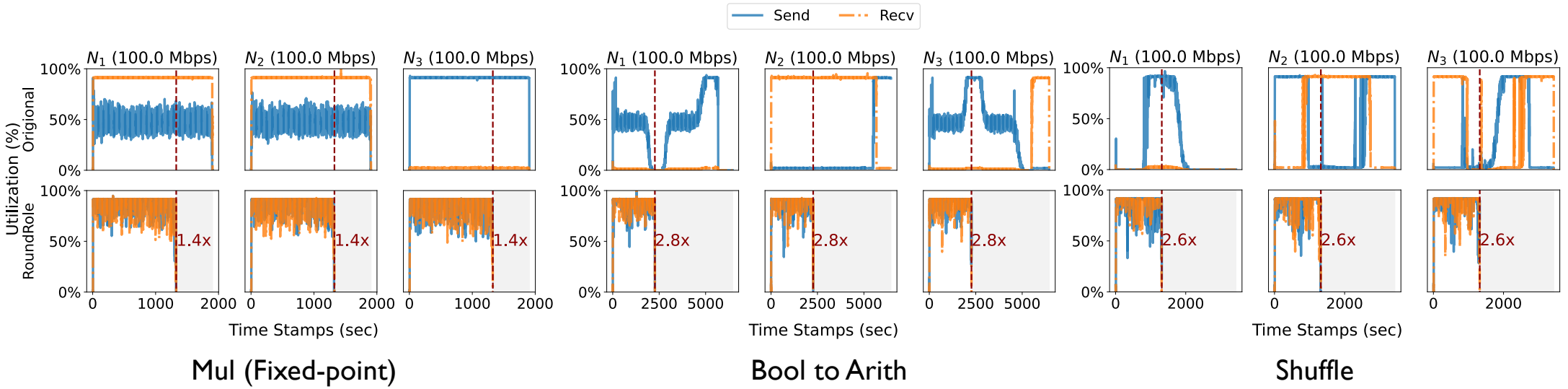


Evaluation: Performance speedup.

- The performance of six protocols with and without RoundRole across six network settings shows significant speedup!



Evaluation: Performance improvement comes from bandwidth utilization.



RoundRole effectively utilizes the originally wasted bandwidth and therefore improving the performance!

Conclusion and future work.

- RoundRole exploits one crucial yet overlooked aspect in practical MPC: the inherently asymmetric communication patterns, then it improves the performance by leveraging the wasted bandwidths.
- Now, MPC researchers can focus on improving the complexity and RoundRole handles all the execution-level details.
- We want to highlight that there optimizing the practical MPC is a multi-angle system, not just the communication rounds and volumes. We should also pay effort on other overlooked problems in the future.

Thanks for your listening!

Authors

For any questions and discussions, feel free to contact fanxy98@mail.tsinghua.edu.cn