# A Dataflow-Oriented Programming Interface for Named Data Networking

Li-Jing Wang, Yong-Qiang Lyu, Ilya Moiseenko,
Dong-Sheng Wang

Wang LJ, Lv YQ, Moiseenko I et al. A dataflow-oriented programming interface for named data networking. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(1): 158–168 Jan. 2018. DOI 10.1007/s11390-018-1812-9

## Research Problem

- Named Data Networking (NDN), as a new data-oriented network architecture, uses data names instead of IP addresses for flowing data.
- NDN enables applications to communicate using Application Data Units (ADU) , which results in great flexibility of application design but also introduces more complex tasks, such as data segmentation, packet verification and flow control.
- *Therefore, the absence of the programming interface in transport layer leads to severe difficulties of NDN application development.*
- In this paper, we propose a dataflow-oriented programming interface to provide a variety of transport-layer strategies for NDN, which greatly improves the efficiency in developing applications.

# Kernel Contributions

- We design a dataflow-oriented programming interface in NDN transport layer. We also implement a video streaming application by utilizing this programming interface to verify its functionality and performance.
- The interface helps application developers handle complex network-layer tasks, such as data segmentation, packet verification, and data retransmission, to guarantee reliable data delivery and improve development efficiency.
- For largely asynchronous publishing, the interface provides a throughput perceptive parallel ADU consumption strategy to achieve high performance. For real-time publishing, the interface provides an adaptive ADU pipeline strategy to control the dataflow based on the current network status and data generation rate.
- The interface also provides network measurement strategies to monitor an abundance of critical metrics that influence application performance.

# Results & Conclusions - Parallel Strategy

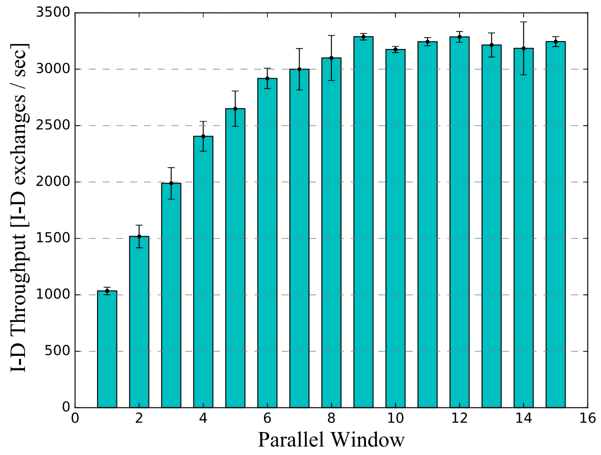

Fig.1. Throughput of Interest-Data exchanges (error bars show 95% CI).

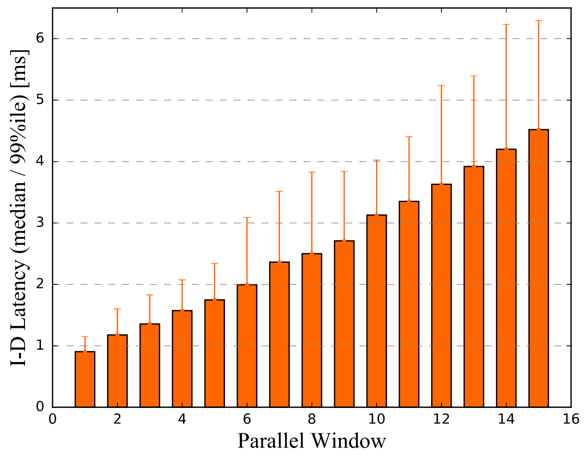

Fig.2. Median latency of Interest-Data exchanges (99%ile indicated by lines atop the bars).

Conclusion1: Parallel Strategy helps retrieve the data with the required throughput by adjusting the parallel window.

Conclustion2: Larger window also brings larger latency.

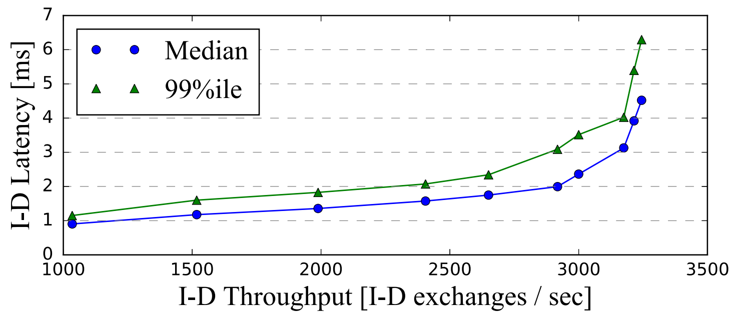

Fig.3. Interest-Data latency vs. throughput.

Conclustion3: The application developer should pay attention to the tradeoff between throughput and latency when use this strategy.

# Results & Conclusions - Pipeline Strategy

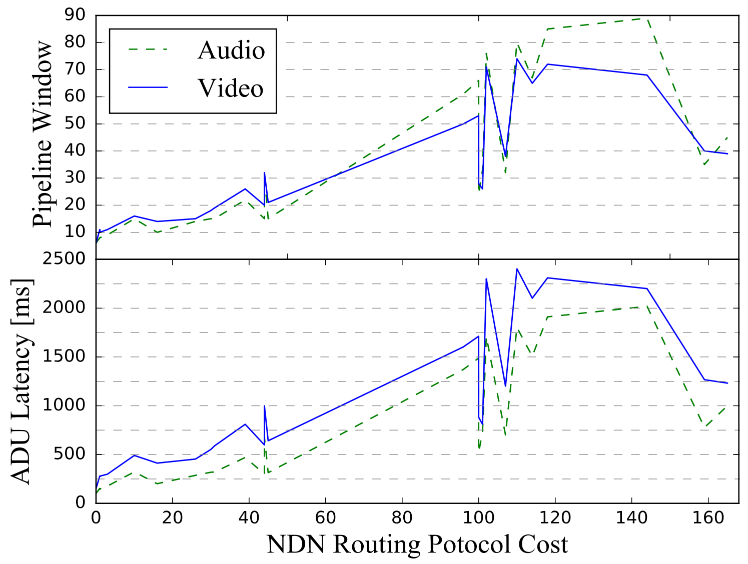

Fig.4. Pipeline window and ADU latency over the world-wide testbed.

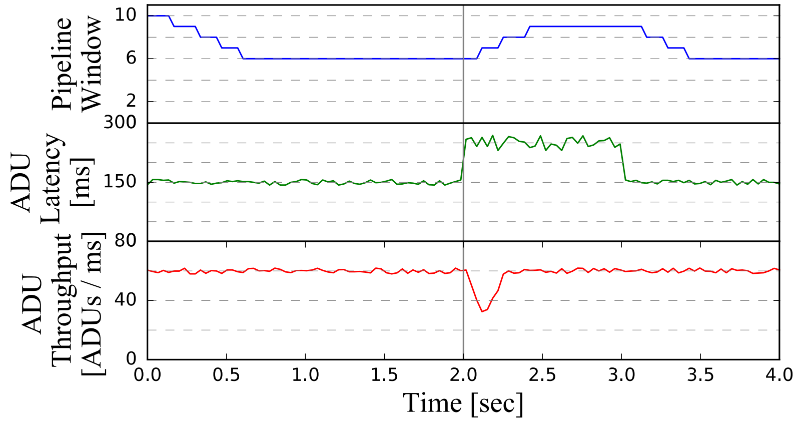

Fig.5. Pipeline window, ADU latency and ADU throughput when network congestion occurs.

Conclusion 4: The growing tendency of ADU latency with respect to NLSR cost verified that the measurement layer can provide authentic network status to the data retrieval strategy.

Conclusion 5: Both the stable throughput for video and audio frames and the similar growing tendency with respect to ADU latency showed that the proper pipeline window and ADU timeout contribute to the stable data retrieval.

Conclusion 6: During the testing period, except for an unavoidable initial playing delay, the video was only slightly stuck (less than 300 ms) at the beginning of congestion due to prolonged I-D exchange latency, and could be kept fluent during the whole progress with stable ADU throughput ≈ 60, which verified that our algorithm can control the dataflow efficiently even facing small network congestions.