Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institute of Computer Science
Chair of Communication Networks
Prof. Dr. Tobias Hoßfeld

Discrete-Time Modeling of NFV Accelerators that Exploit Batched Processing

Stanislav Lange, Leonardo Linguaglossa, **Stefan Geissler**,
Dario Rossi, Thomas Zinner

comnet.informatik.uni-wuerzburg.de

TELECOM ParisTech

Technische Universität Berlin

SENDATE PLANETS

NFV Advantages

Service Elasticity

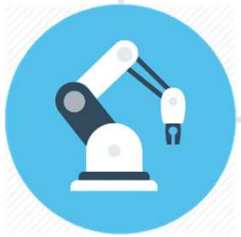
- Dynamic scaling
- Virtualization
- Reduced complexity in high availability scenarios

Financial Benefits

- OPEX decrease
- CAPEX reduction through COTS hardware

Network Automation

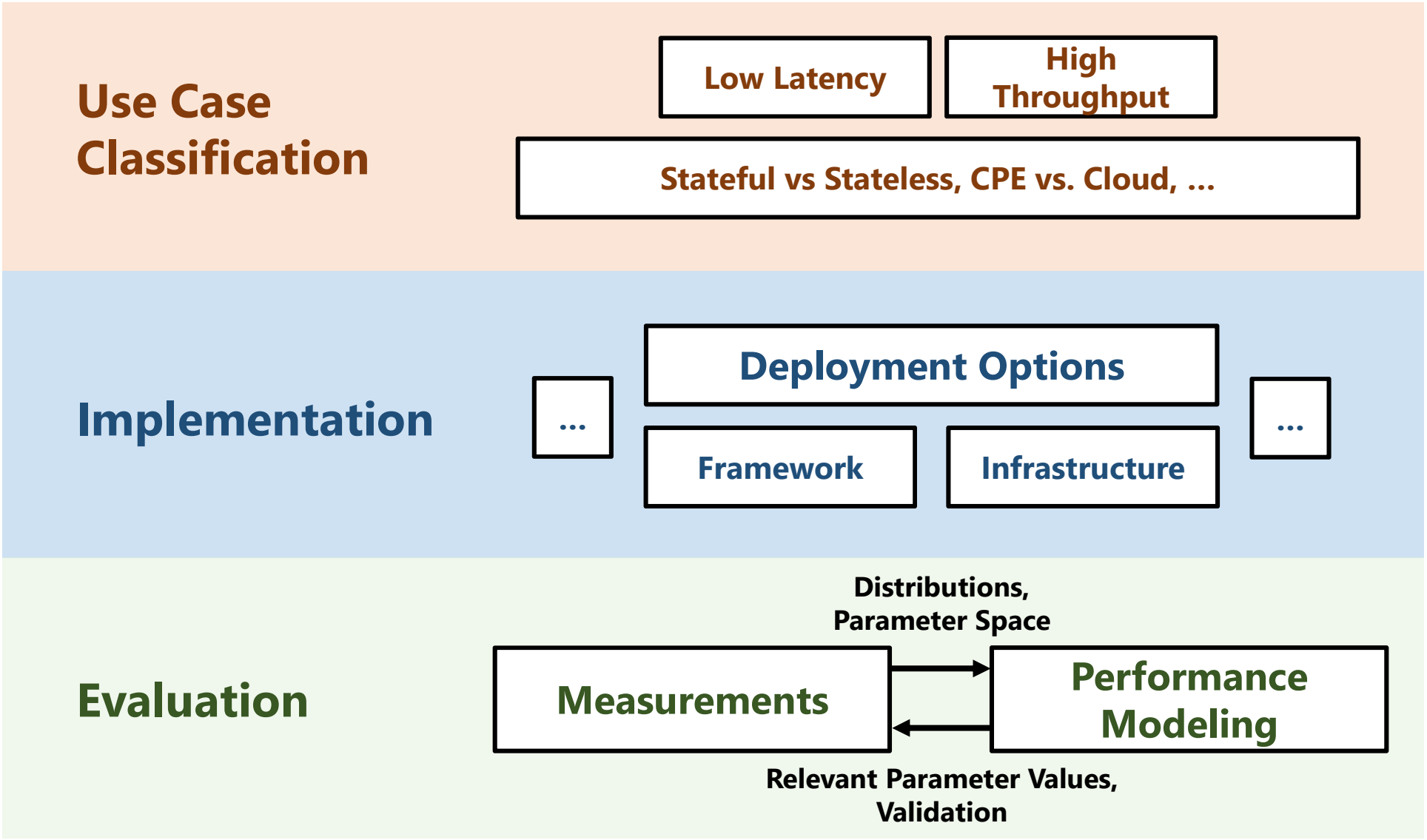
- Network programmability
- Increased flexibility
- Improved interoperability



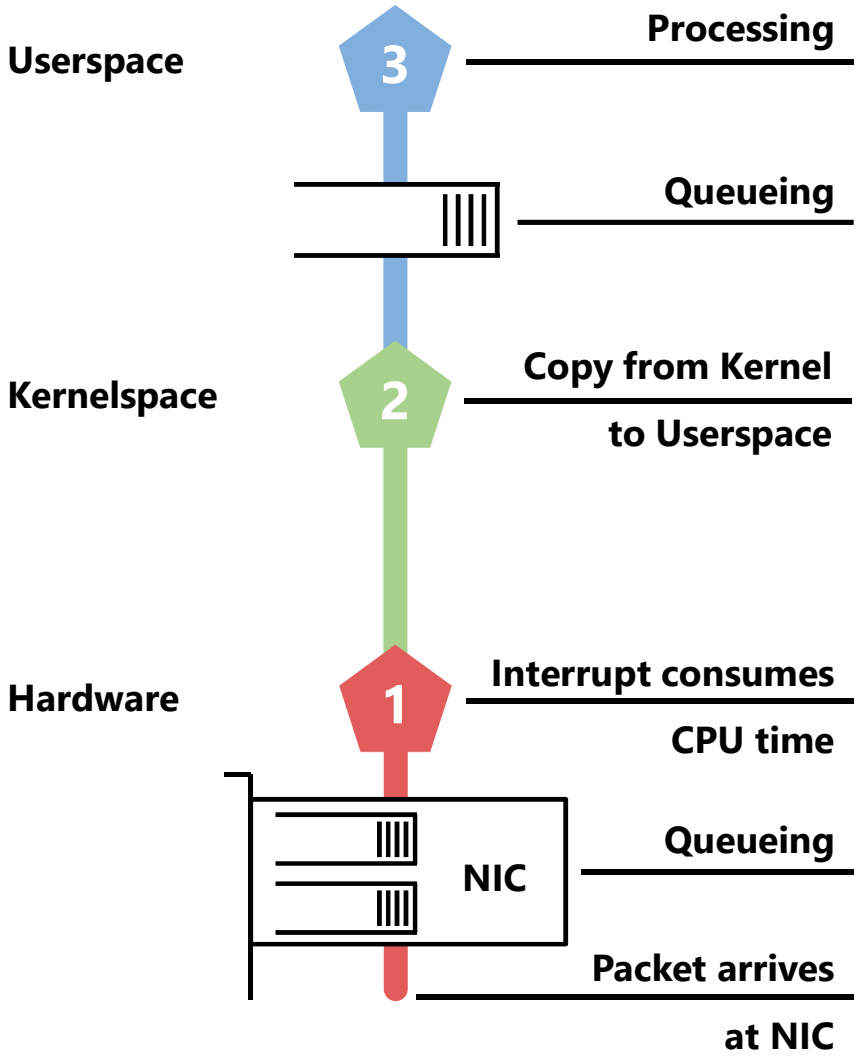
Optimization Potential

- Dynamic placement
- Continuous reoptimization
- Fast redeployment and rollout

NFV Considerations



Traditional NAPI-based I/O

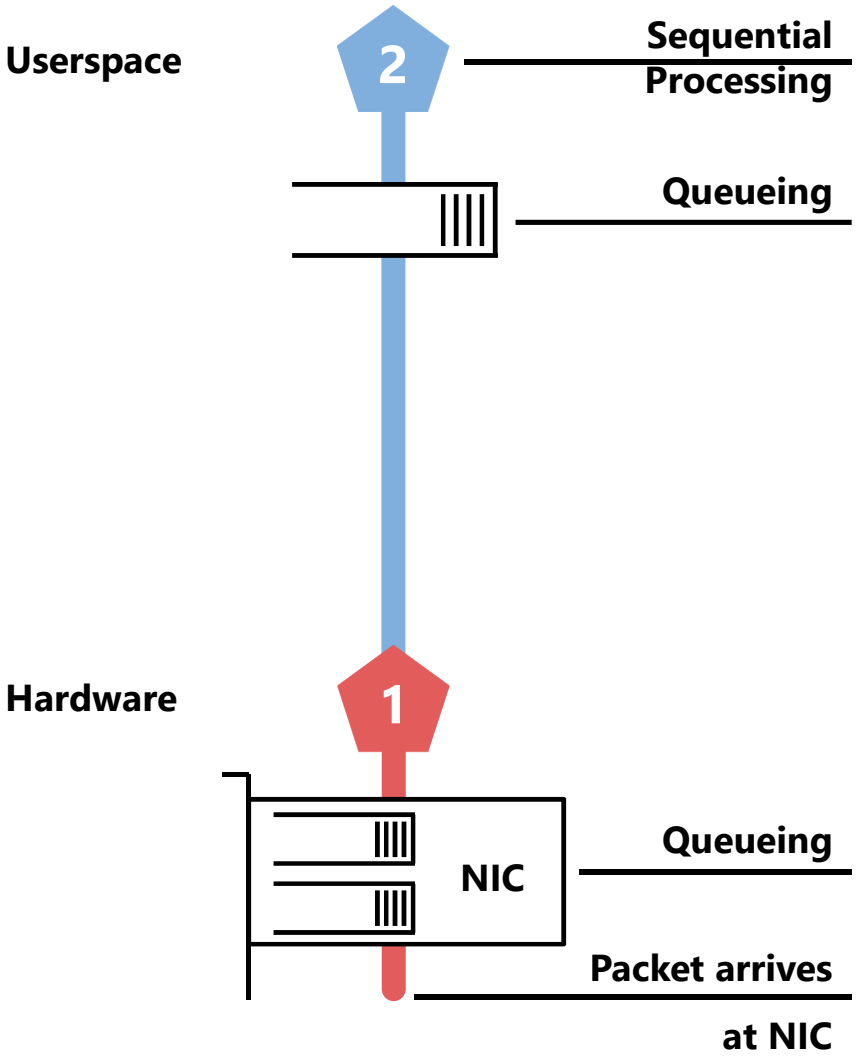


- Packets are **queued** for processing
- Packets are processed **sequentially**

- Packets are copied to Kernelspace
- Headers are processed
- Packets are copied to Userspace

- Arriving packets are **queued** at NIC
- **Interrupt mitigation** mechanisms reduce interrupt load

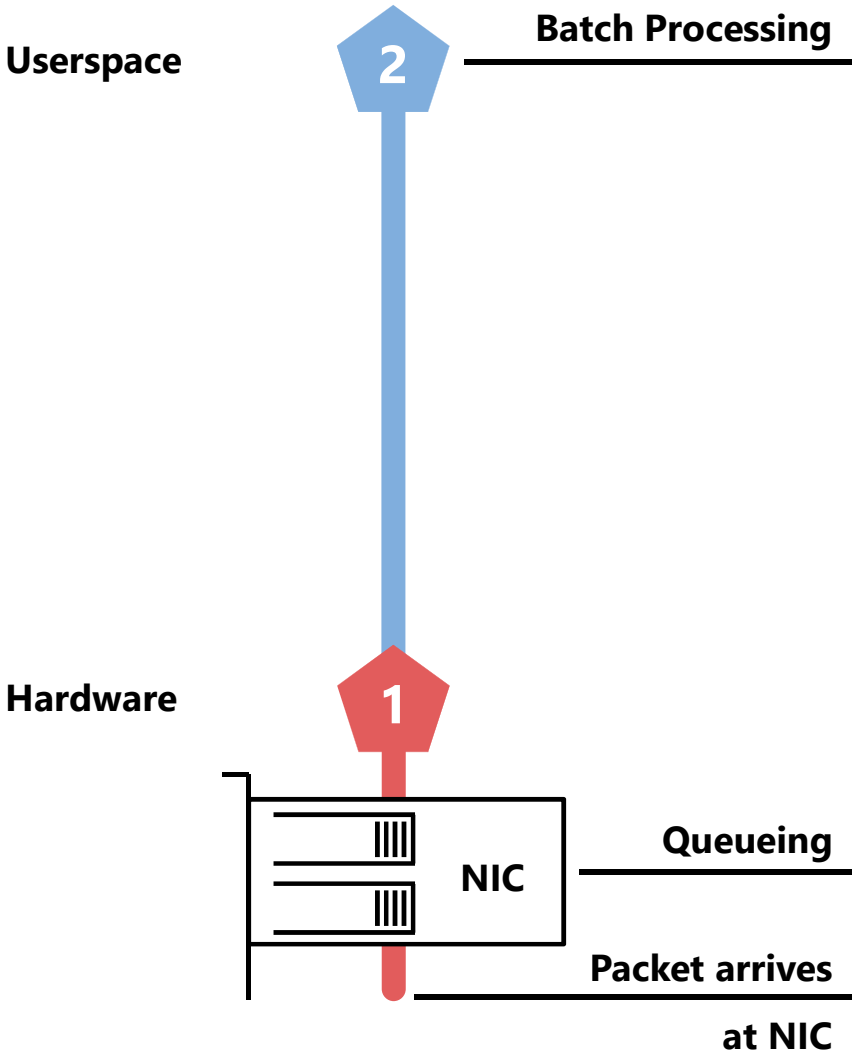
Kernel Bypass and I/O Batching



- Userspace application collects packets directly from NIC
- Dedicated CPU-core for active **busy polling**
- Packets are processed **sequentially**

- Arriving packets are **queued** at NIC
- **No interrupts** are triggered for arriving packets

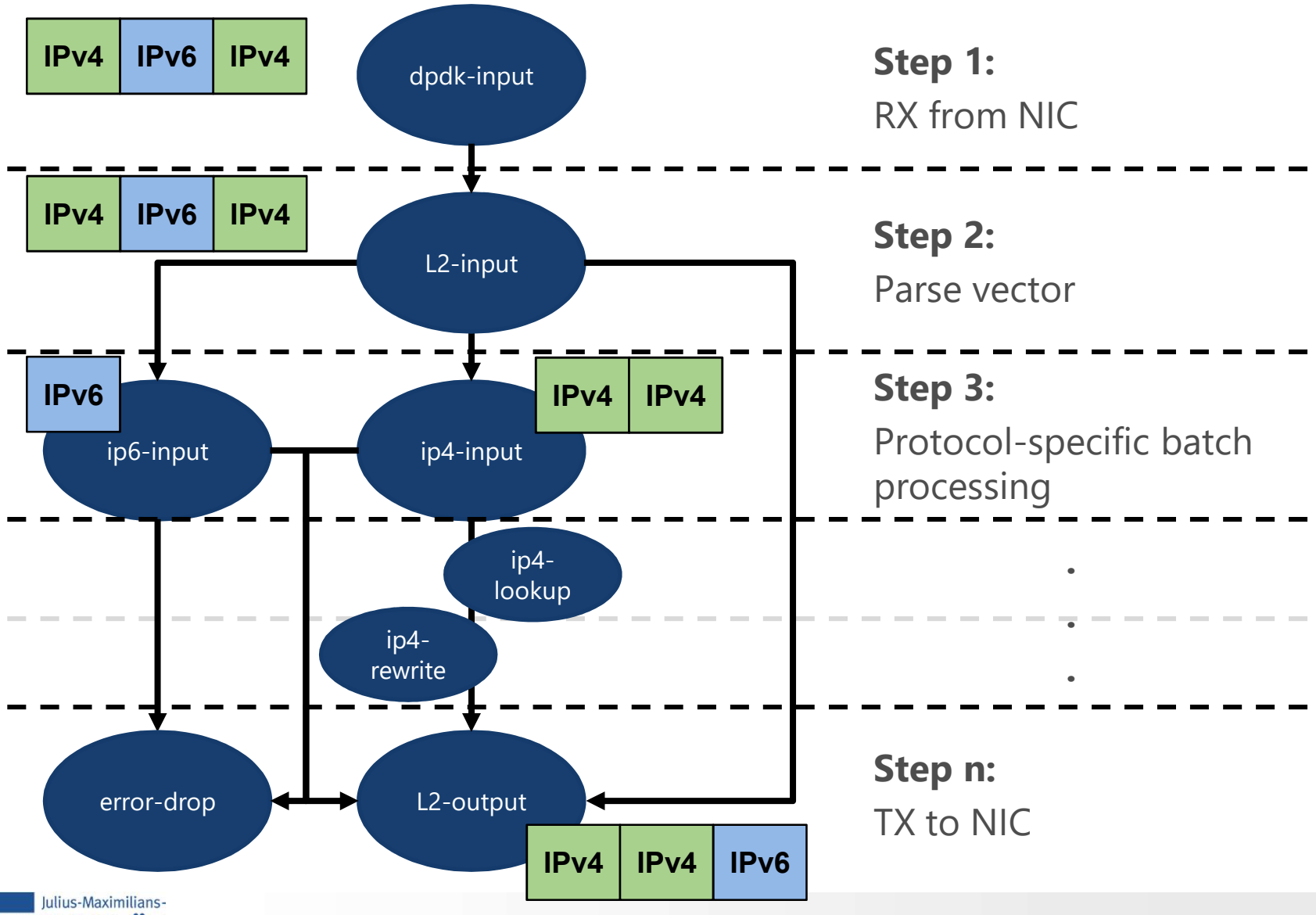
Kernel Bypass and Compute Batching



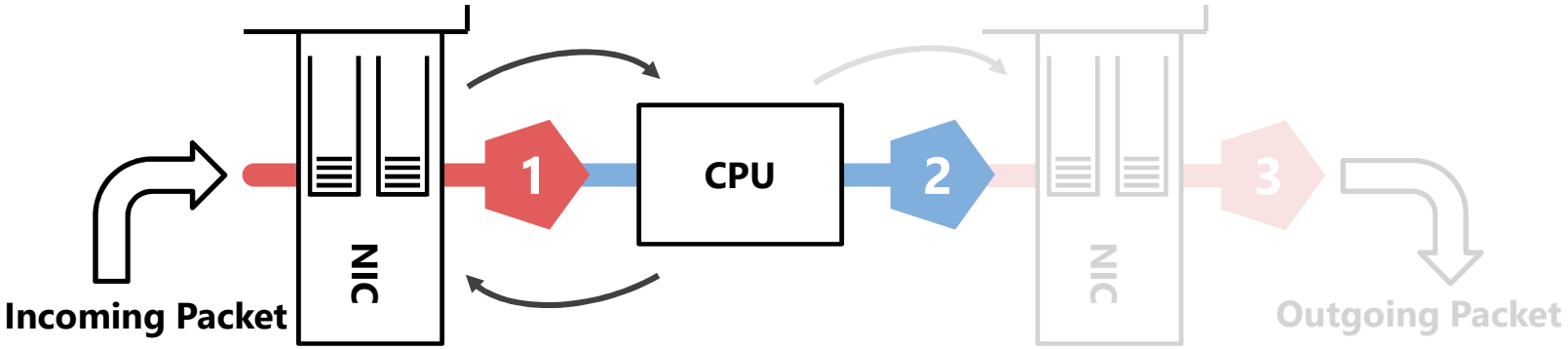
- Userspace application collects packets directly from NIC
- Dedicated CPU-core for active **busy polling**
- **Batch processing** of packets in form of packet vectors

- Arriving packets are **queued** at NIC
- **No interrupts** are triggered for arriving packets

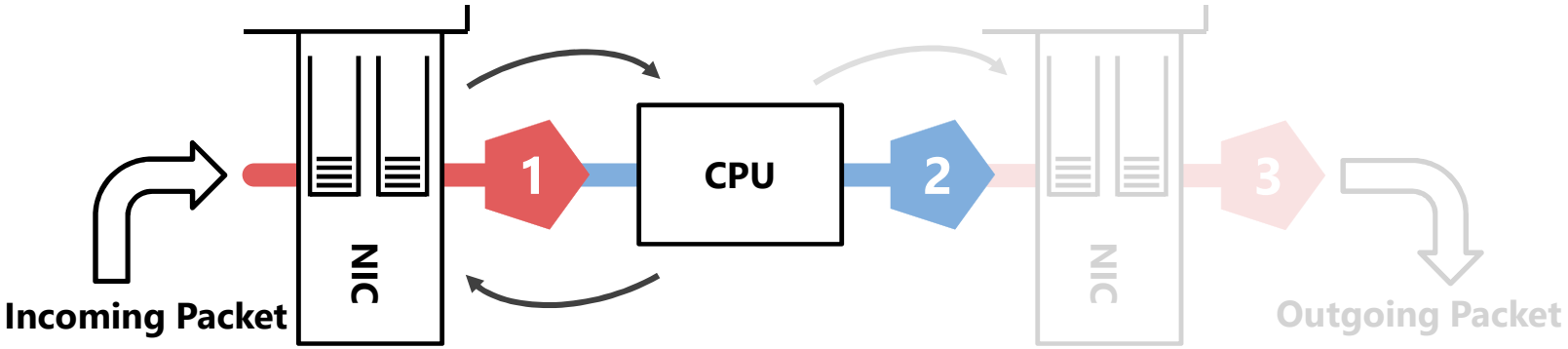
Vector Packet Processing (VPP)

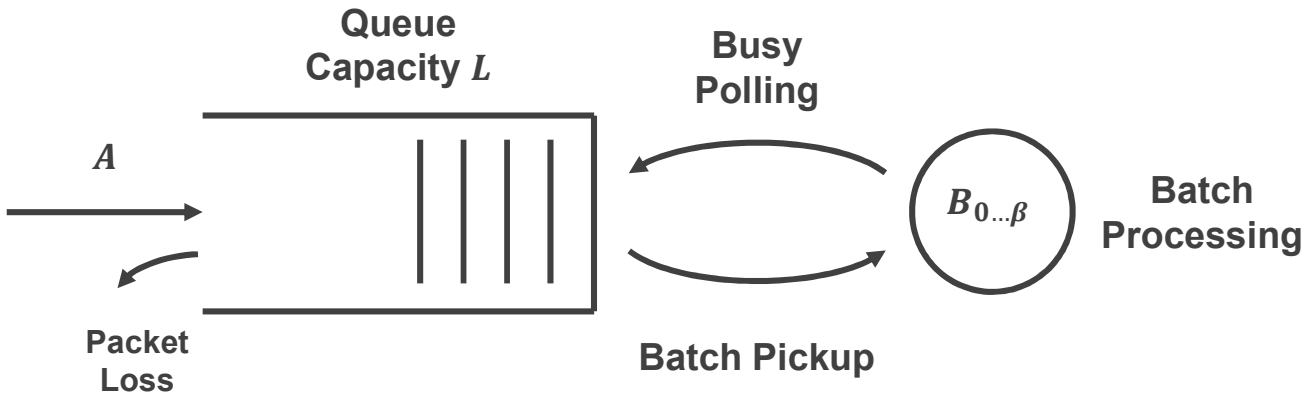


System Abstraction



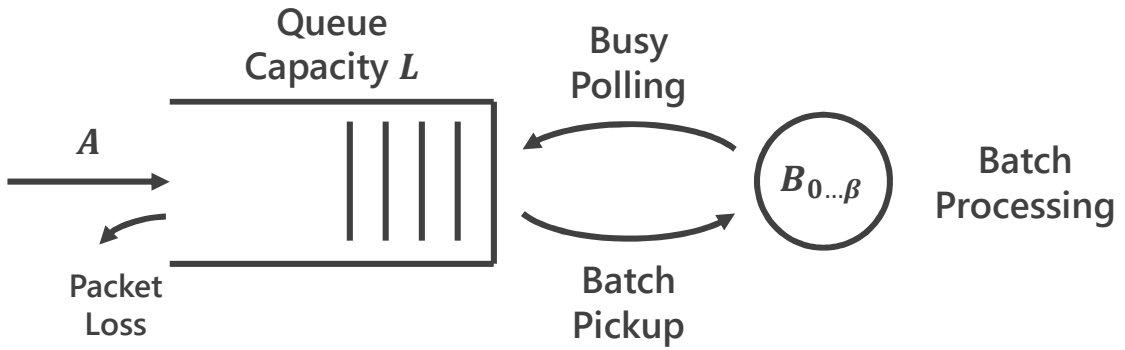
System Abstraction





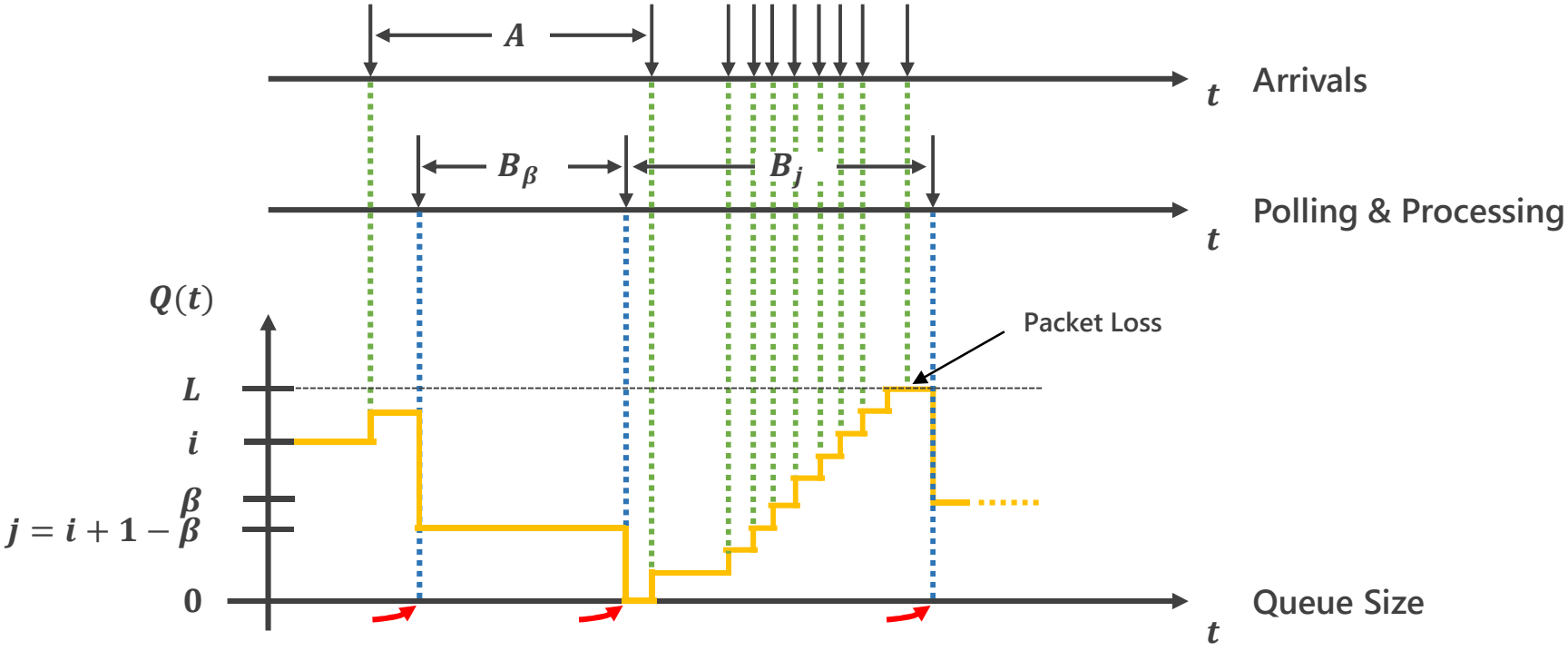
Discrete-Time Queueing Model

- Inputs
  - Distribution of packet interarrival times A
  - Distribution of size-dependent batch service time B_i
  - Queue capacity L, maximum batch size β



- Outputs
  - Batch size **distribution**
    - → Efficiency indicator
  - Packet loss probability
    - → Identification of **operational regimes**

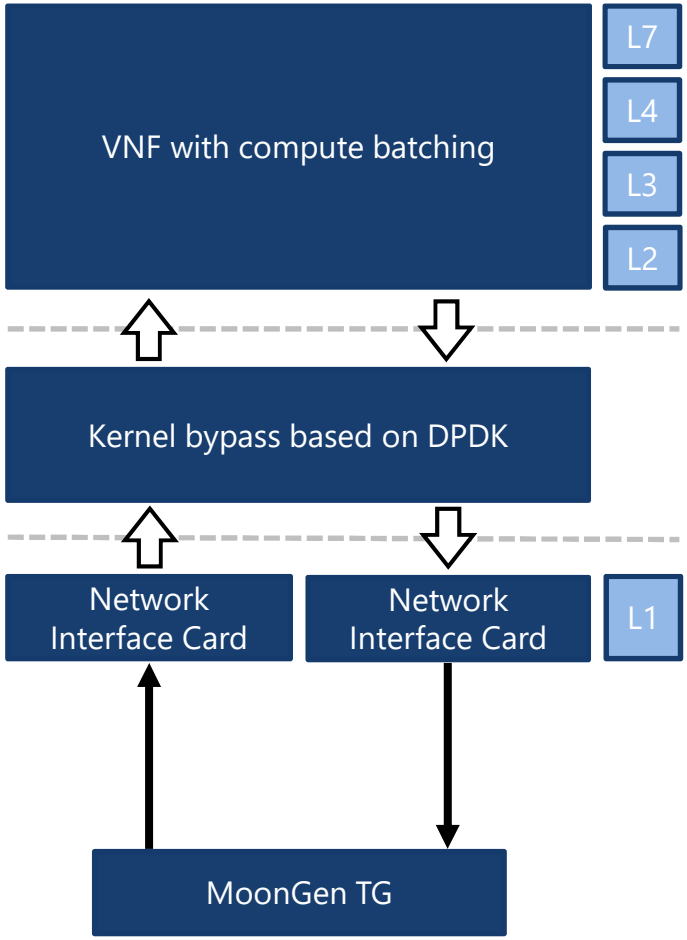
Embedded Markov Chain



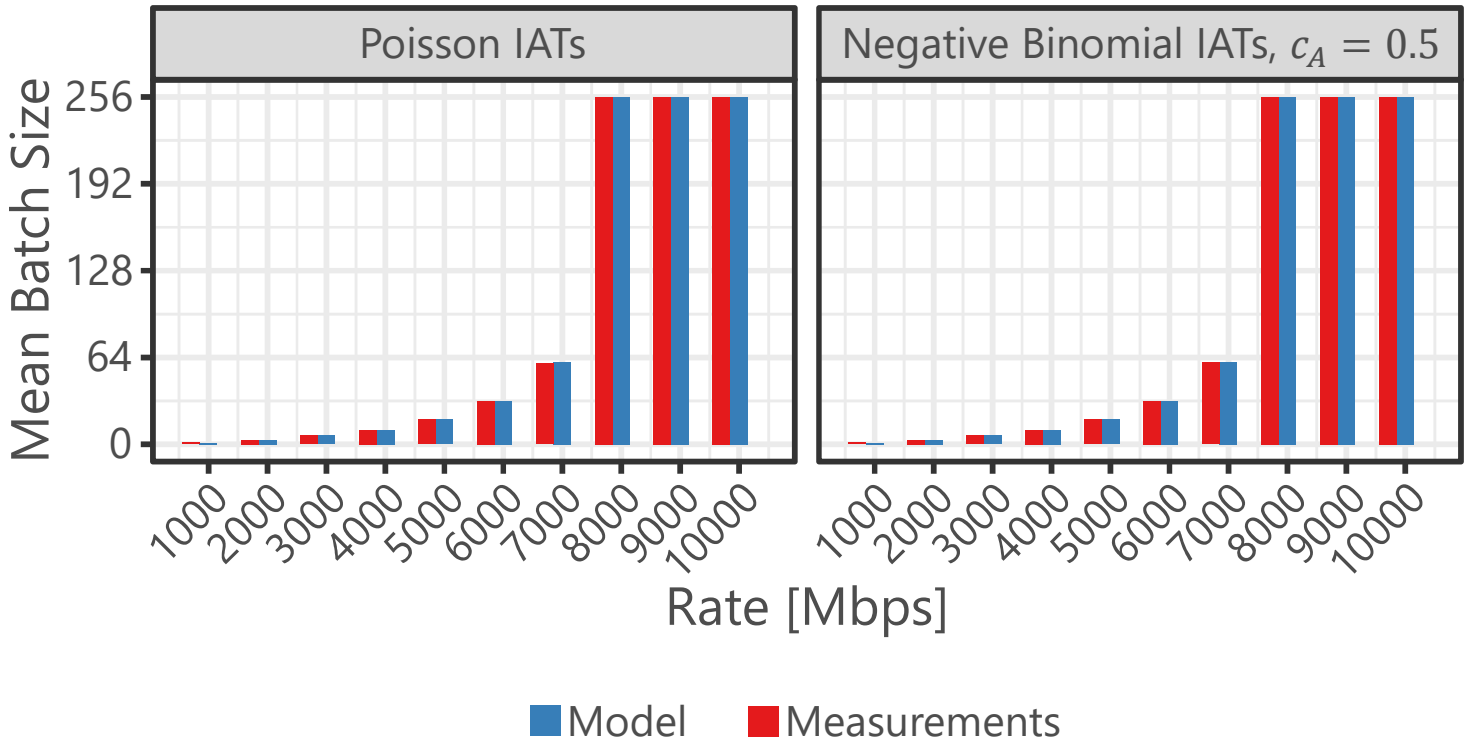
- Markov Chain with **embedding times** right before polling events
  - Solved through **fixed point iteration**
  - Allows computation of **queue size distribution at embedding times** and **batch size distribution**

Experimental Testbed

- **VPP based VNF**
  - Cross connect (XC)
  - Ethernet, IPv4, IPv6
- **Network Stack**
  - Compute batching using VPP
  - I/O batching using DPDK
- **MoonGen** software traffic generator
  - Generates up to 10G traffic
  - Monitors end to end delay and packet loss



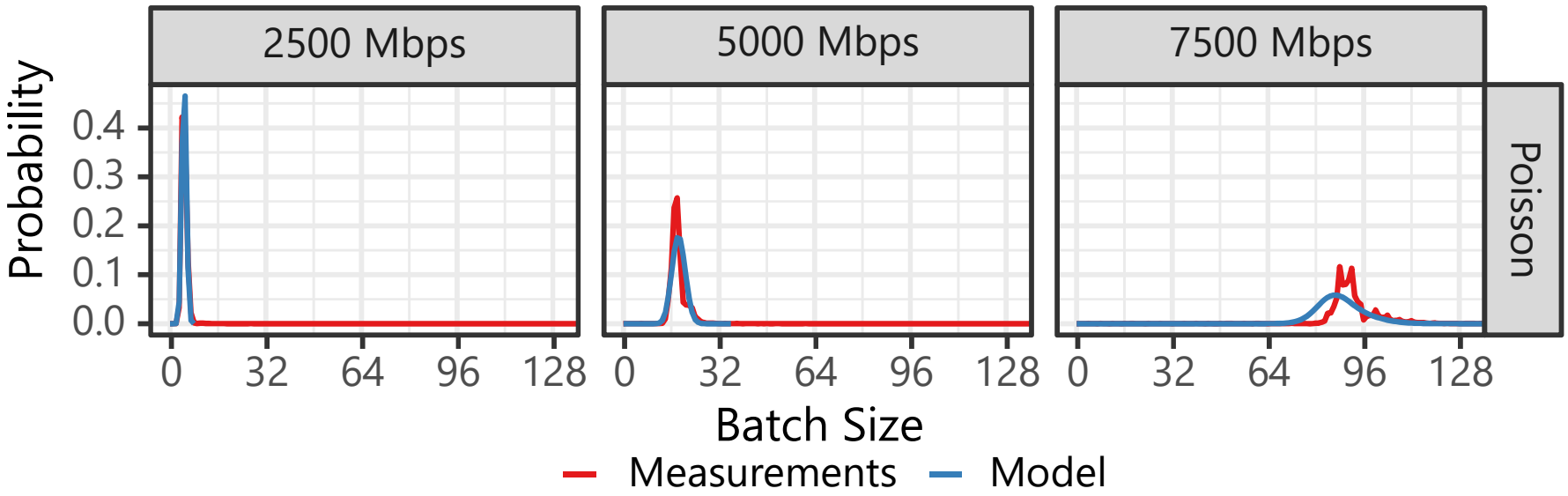
Cross Connect Scenario - Mean Batch Size



- Mean batch size is **robust** w.r.t. arbitrary packet arrival processes

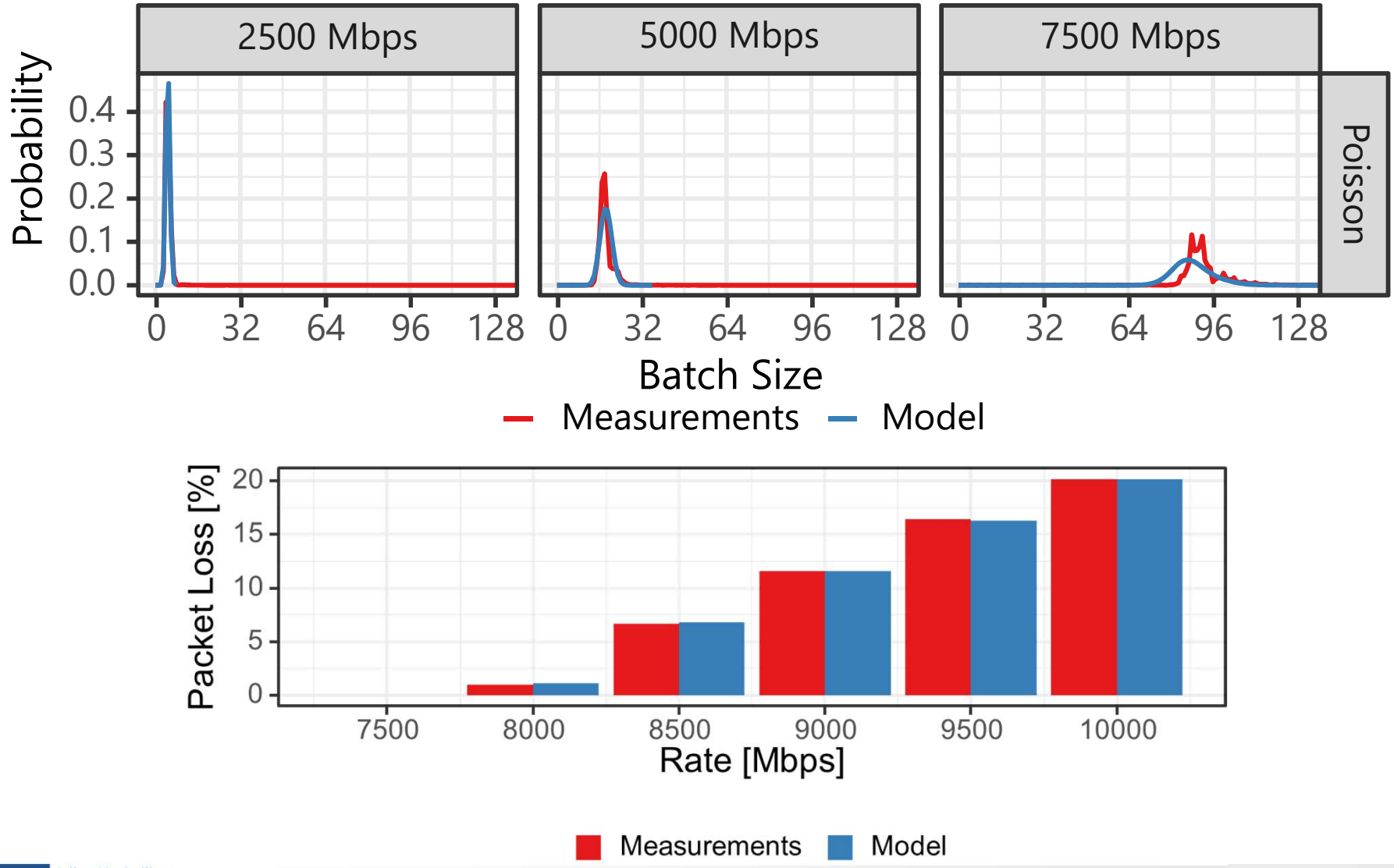
➔ But what about the distribution?

Cross Connect Scenario – Batch Size Distribution

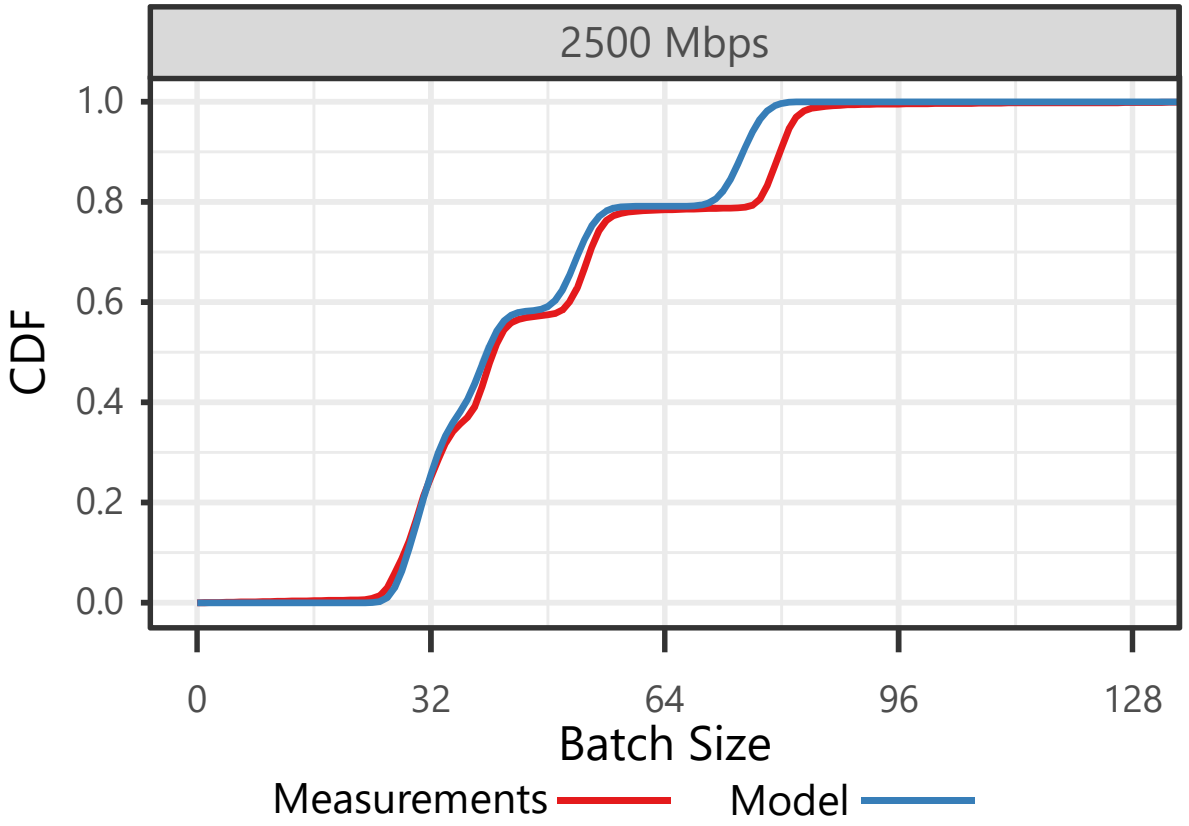


- Model accurately predicts entire distribution
  - Allows in-depth assessment of **VNF efficiency**
- Rate-dependent peaks indicate **equilibrium** between batch service time and number of arrivals

Cross Connect Scenario – Batch Size Distribution



Mixed Traffic Scenario – Batch Size Distribution



- **Unmodified** model achieves high accuracy even in complex scenario
  - **General** model
- Systematic mismatch caused by implementation details

Conclusion

- Discrete-time **queueing model** for **batched packet processors**
  - Solved via numerical **fixed point iteration**
  - Allows assessment of **queue size and batch size distributions** as well as **packet loss probability**

- Validation with **VPP-based VNF** in simple cross-connect scenario as well as with **mixed traffic**
  - Model **generalizes well**, achieving **close fit** in both scenarios
  - Enables accurate **performance prediction**

- Possible **future extensions** encompass
  - Validation with **other frameworks** like FastClick or G-Opt
  - Addition of **further KPIs** like waiting time distribution and jitter
  - Evaluation of **more complex traffic patterns**

… math …

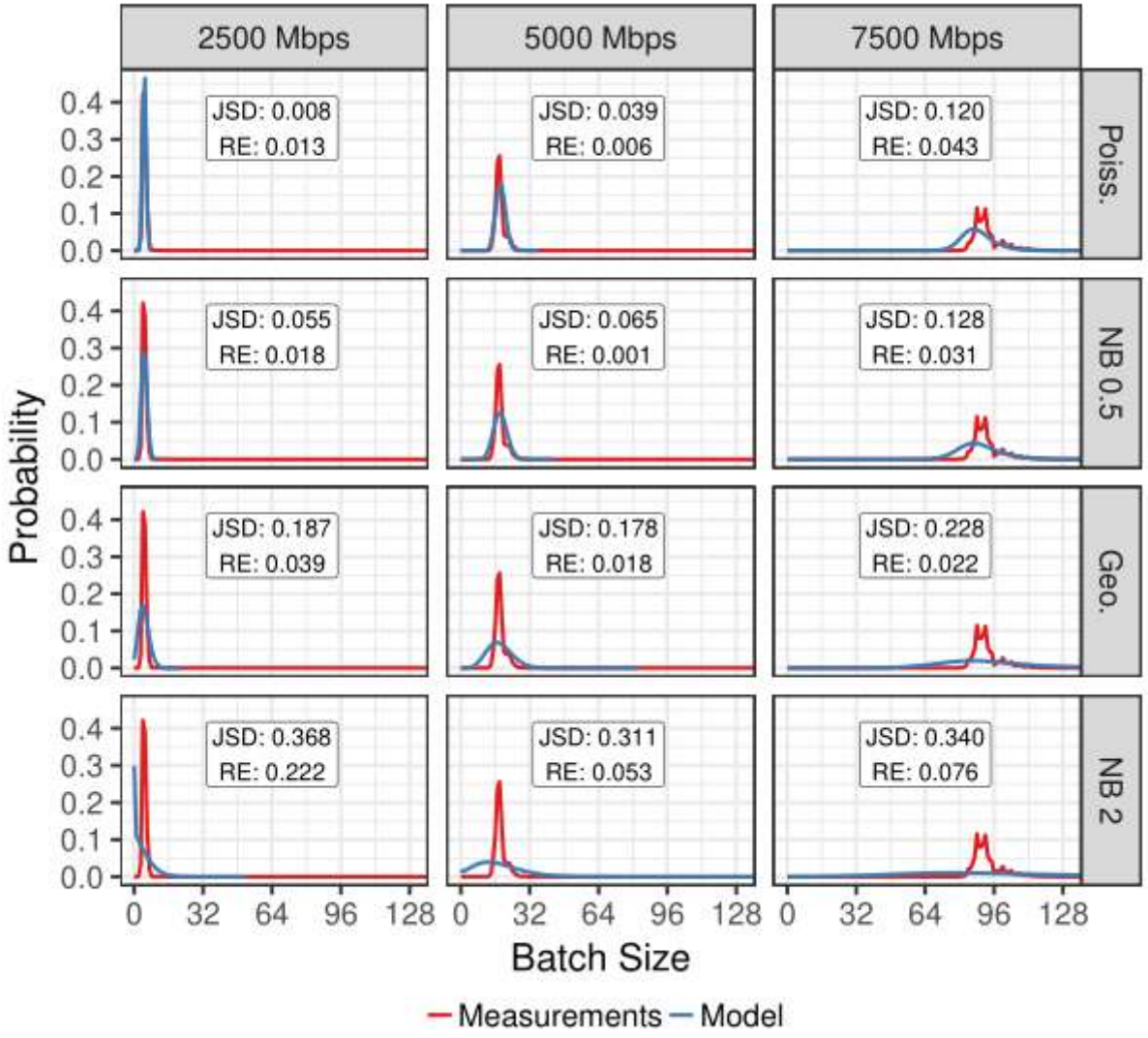
$$q_{n+1}(k) = \begin{cases} \sum_{i=0}^{L} q_n(i) x_{b_{\min(i,\beta)},a}(k - (i - \min(i,\beta))) \\ \quad \text{for } k < L, \\ \sum_{i=0}^{L} q_n(i) \sum_{j=0}^{\infty} x_{b_{\min(i,\beta)},a}(L + j - (i - \min(i,\beta))) \\ \quad \text{for } k = L, \\ 0 \text{ otherwise.} \end{cases}$$

$$v(k) = \begin{cases} q(k) & k < \beta, \\ \sum_{i=\beta}^{\infty} q(i) & k = \beta, \\ 0 & \text{otherwise.} \end{cases}$$

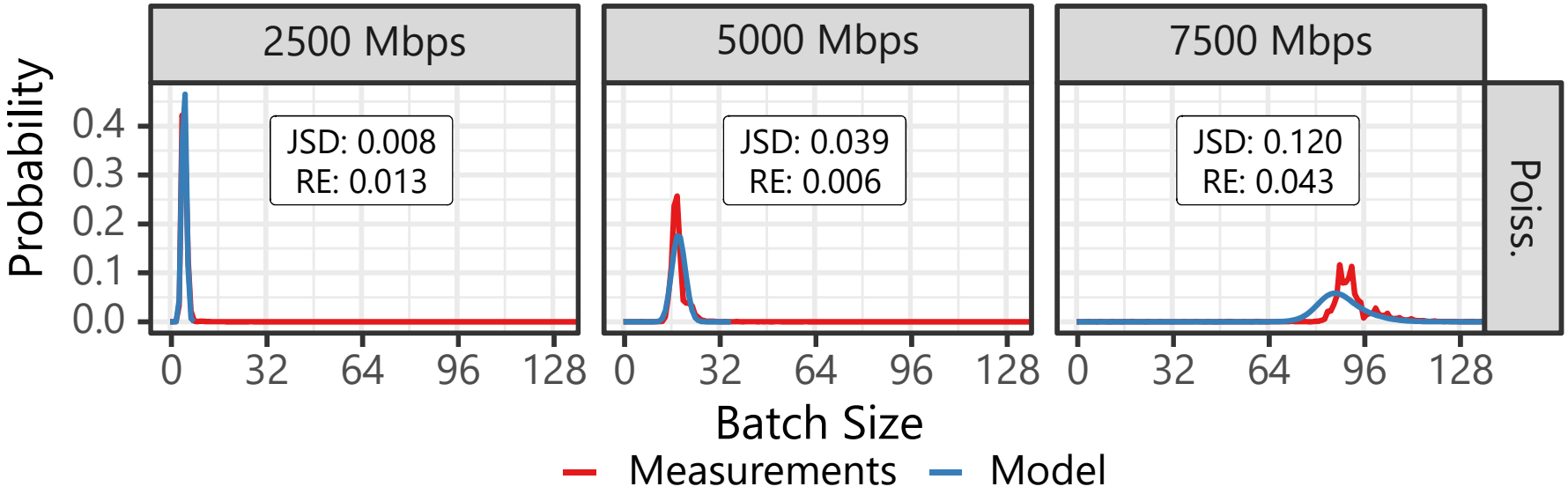
$$x(j) = \tau(0)\,\delta(j) + \sum_{m=1}^{\infty} \tau(m) \sum_{i=0}^{m-1} \left(f^{(j)}(i) - f^{(j+1)}(i) \right), \quad j = 0, 1, \ldots$$

$$p_{loss} = \frac{\sum_{i=0}^{L} q(i) \sum_{j=0}^{\infty} j\, x_{b_{\min(i,\beta)},a}(L + \min(i,\beta) - i + j)}{\sum_{i=0}^{L} q(i) \sum_{j=0}^{\infty} (j\, x_{b_{\min(i,\beta)},a}(j))}$$

Computation of Batch Size Distribution

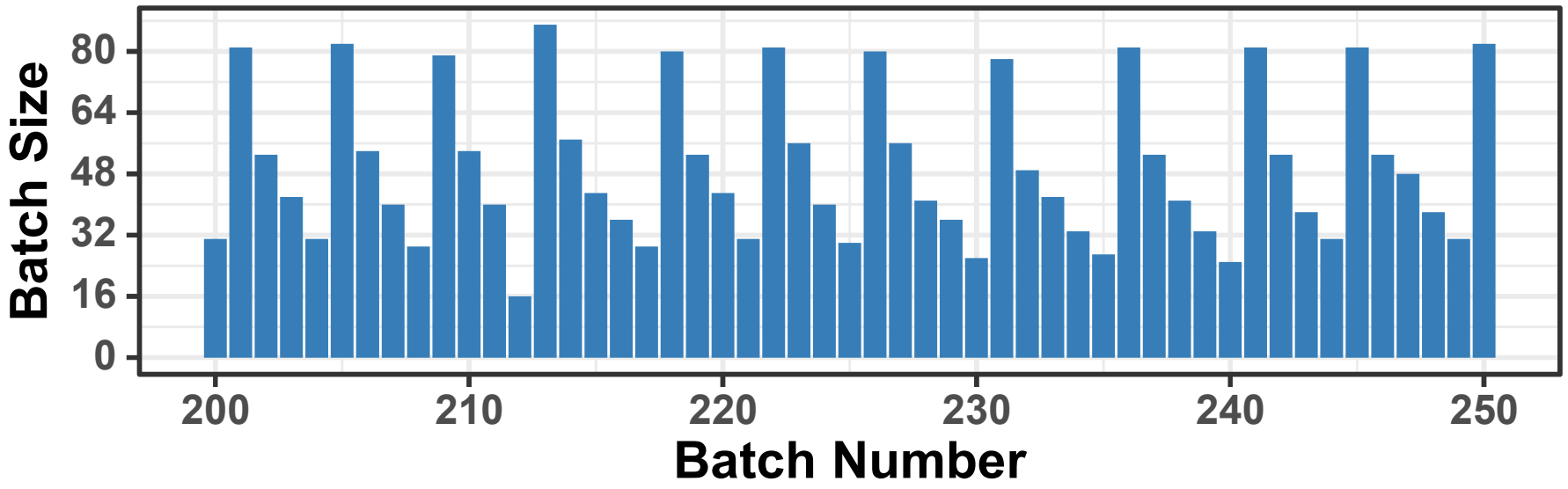


Cross Connect Scenario – Batch Size Distribution

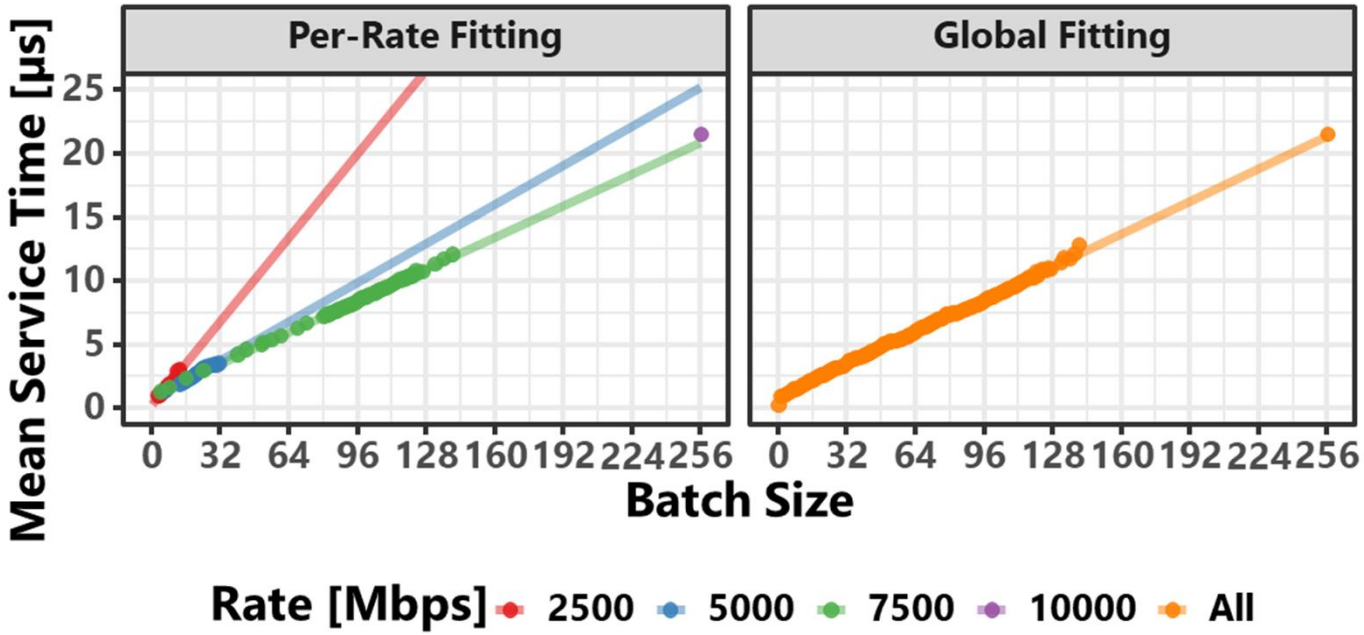


- Rate-dependent peaks indicate **equilibrium** between batch service time and number of arrivals
- Model accurately predicts entire distribution
  - Allows in-depth assessment of **VNF efficiency**
- Closest match when using Poisson-distributed packet interarrivals
  - Reflects behavior of software-based CBR traffic generator

Batch size over time



Model Input via Measurements



- **Mean service time** for different **batch sizes** obtained via measurements at different **load levels**
- Linear fit to obtain mean service times for **all possible** batch sizes
  - Global fit vs. per-rate fit