Quality/Latency-Aware Real-Time Scheduling of Distributed Streaming IoT Applications

Kamyar Mirzazad Barijough, **Zhuoran Zhao**, **Andreas Gerstlauer**

System-Level Architecture and Modeling (SLAM) Lab

Department of Electrical and Computer Engineering

The University of Texas at Austin

https://slam.ece.utexas.edu

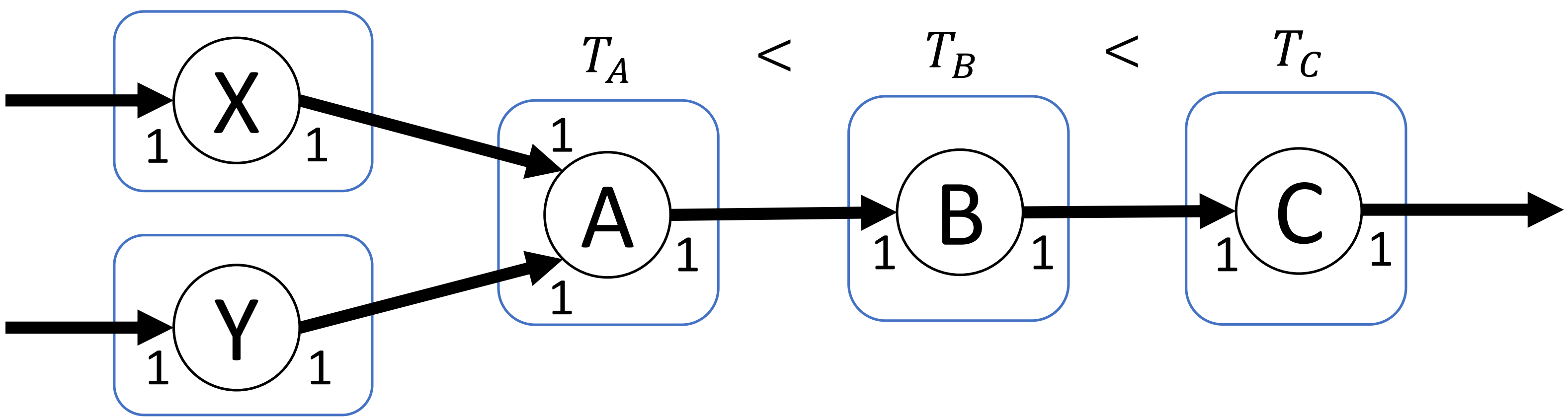
The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Background

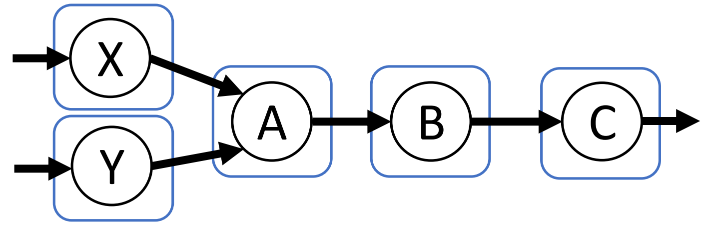
- **Networked and distributed embedded systems**
  - Internet of Things (IoT) and edge computing
  - Networked cyber-physical systems (CPS)
  - ➢ Distributed embedded computing
- **Real-time guarantees**
  - Interact with physical world
  - ➢ Hard latency requirements
- **Open networks**
  - Dynamically changing traffic sources & patterns
  - ➢ Non-deterministic and potentially unbounded latency
- ➢ **Key challenge**
  - ➢ How to provide real-time guarantees over unpredictable networks?

Motivation

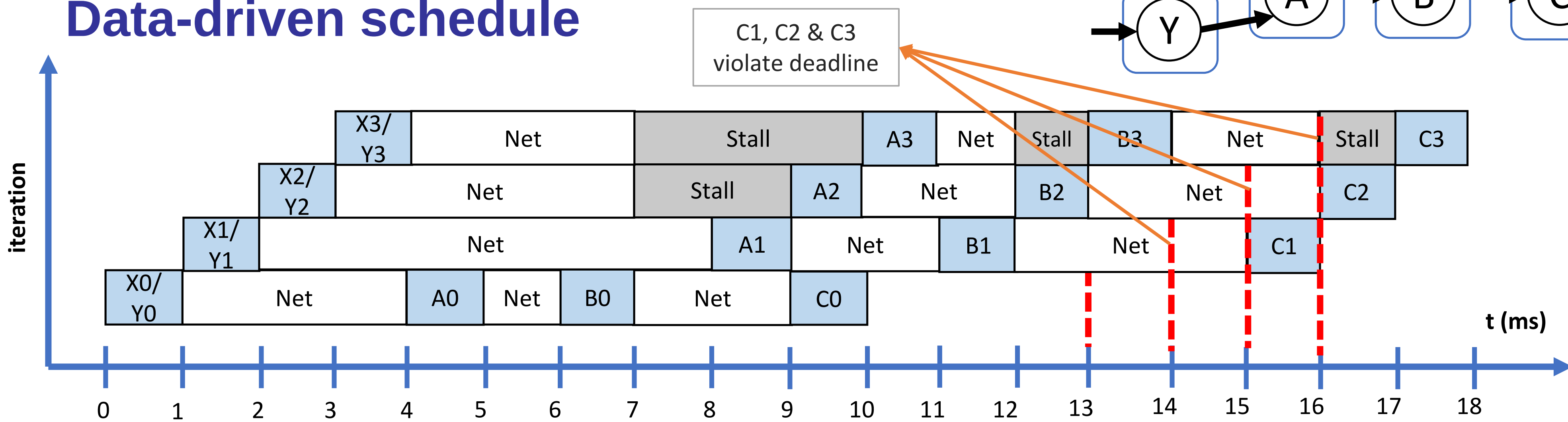
- **Embedded applications are often of streaming nature**
  - Best expressed as a data flow graph
- **Latency guarantees provided via timeouts**
  - Tradeoff between latency and losses (quality)
  - Per-actor timeouts
- ➢ **Timeout assignment for distributed real-time data flow**
  - Partition latency budget across nodes
  - Application/network-dependent tradeoff



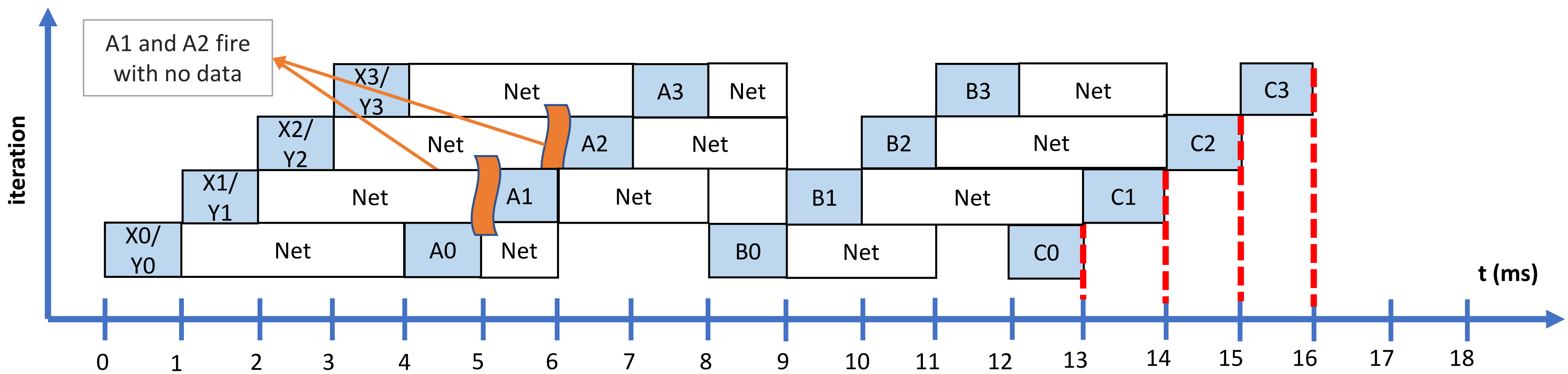
Latency Budget Assignment



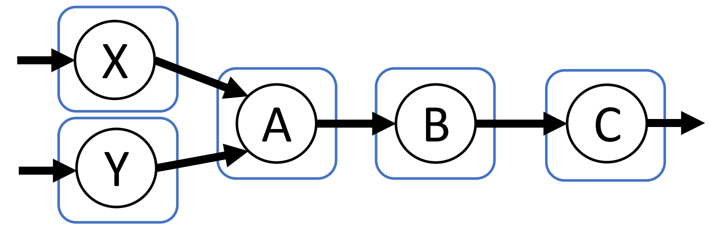
Data-driven schedule



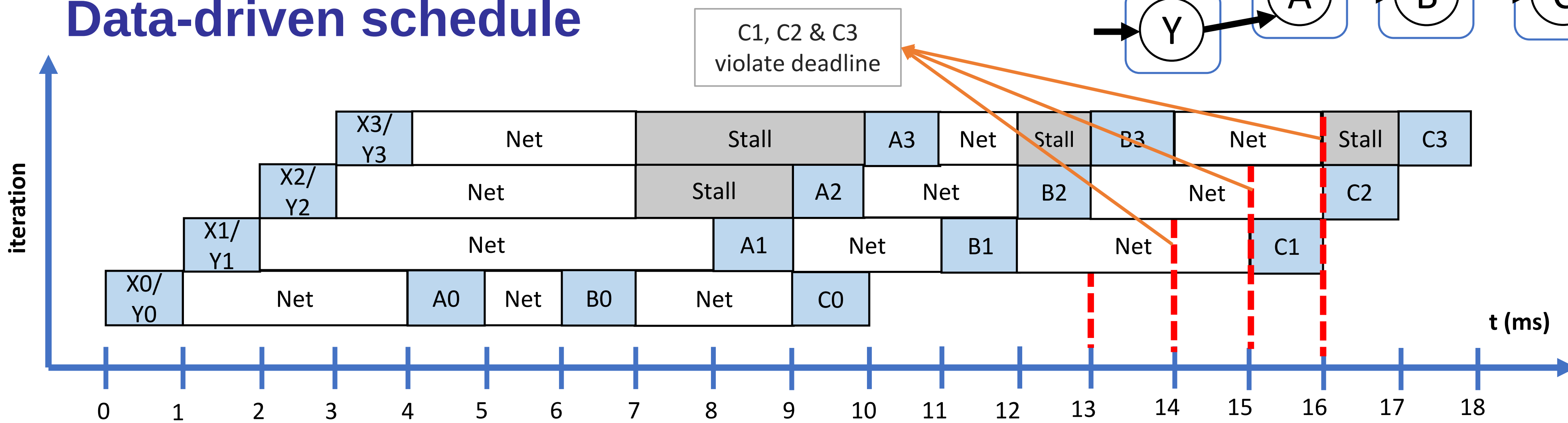
Schedule with timeouts & uniform latency distribution



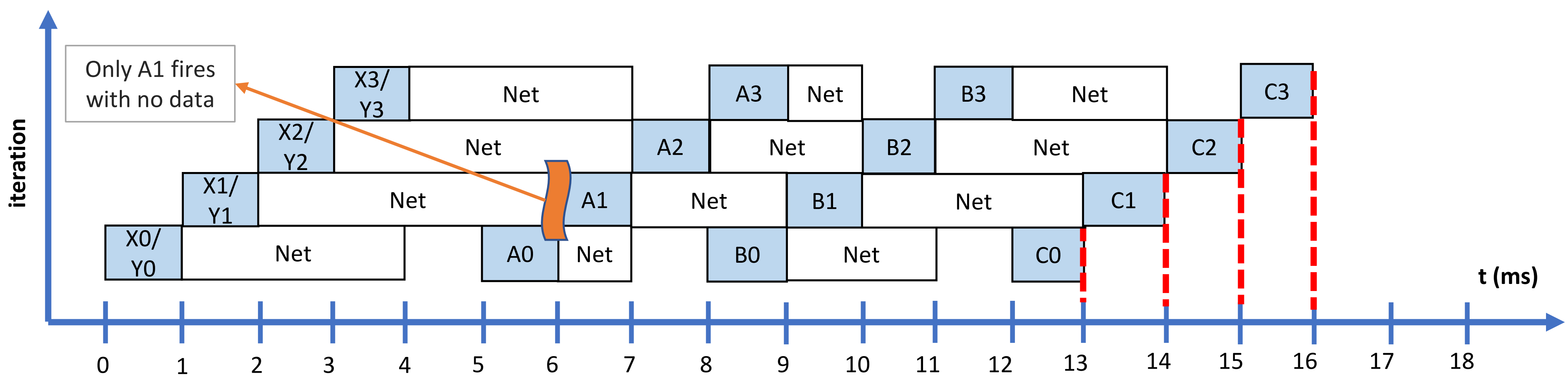
Latency Budget Assignment



Data-driven schedule



Schedule with optimized latency budget distribution



Related Work

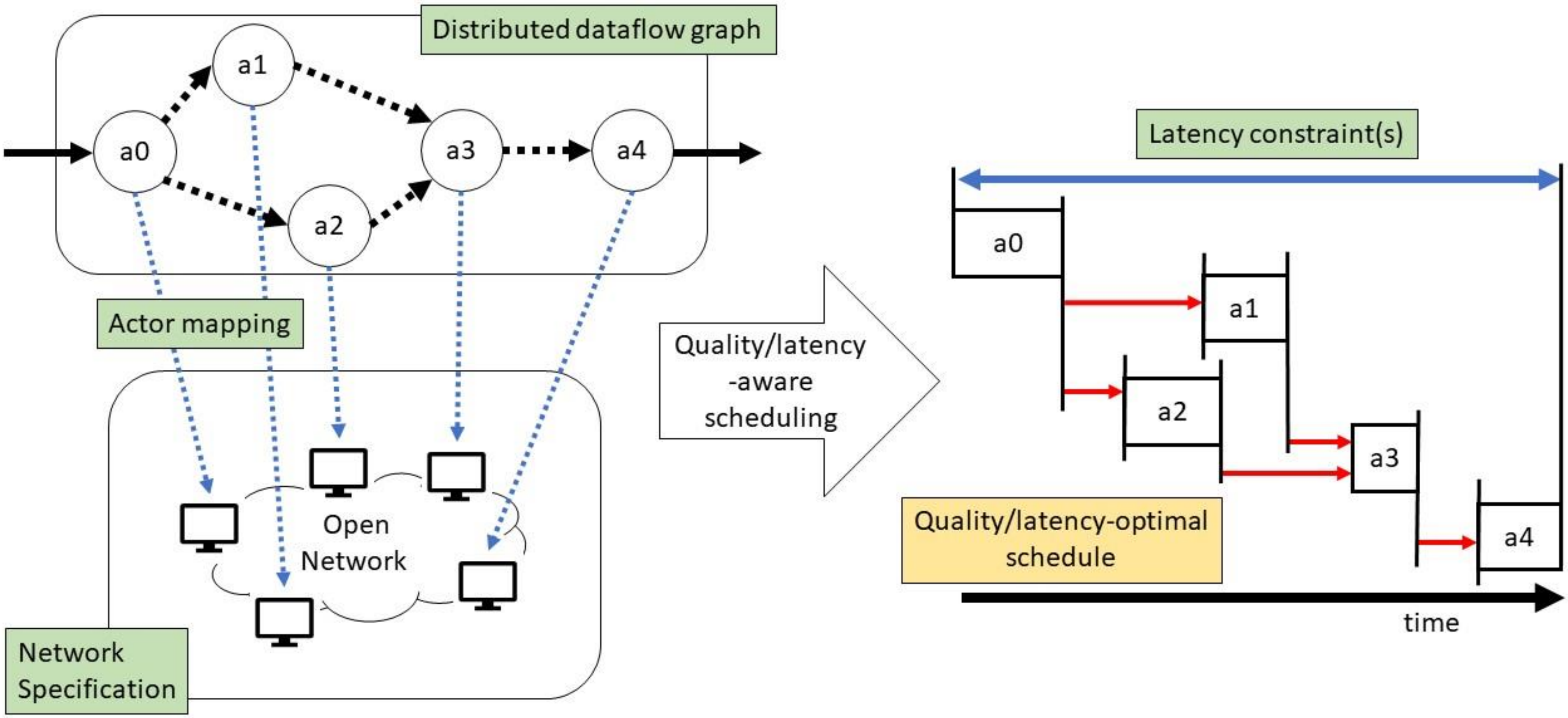
- **Real-time transfer protocol (RTP) [Schulzrinne’03]**
  - Designed for end-to-end data transfer
  - ➢ Only pair-wise/end-to-end timeout assignment

- **Distributed real-time computing frameworks**
  - Real-time extension(s) of RPC frameworks [RT-Corba]
  - Stream processing frameworks [Typhoon, Ares, Storm]
  - ➢ Requires QoS guarantees or reliable delivery from network

- **PTIDES [Zhao’07]**
  - Discrete-event execution for distributed systems
  - ➢ Requires accurate time synchronization and bounded network delay

- **Reactive and Adaptive Data Flow Model (RADF) [Francis’17]**
  - Dataflow with extensions for modeling network effects
  - ➢ No timeout assignment/implementation

Overview



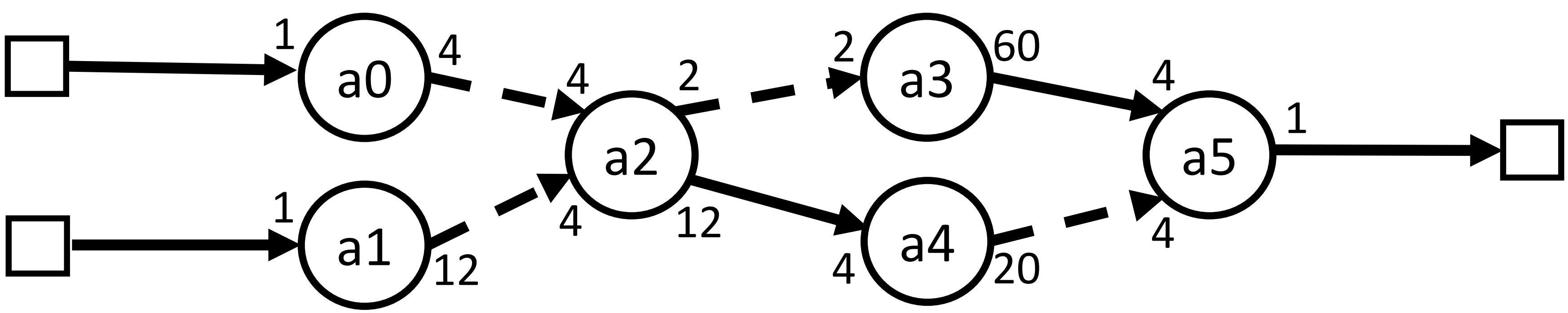
- **Derive a schedule for the given data flow graph using**
  - Worst-case execution times (WCET)
  - Mapping information
  - Latency constraints
  - Network specification

Outline

- ✓ **Introduction**
  - ✓ Motivation, background
  - ✓ Related work
- **Formalizing distributed data flow**
  - Timed extension of RADF
- **Scheduling distributed data flow**
  - Quality model and optimization
- **Experimental Results**
- **Summary & Future Work**

Reactive and Adaptive Data Flow (RADF)

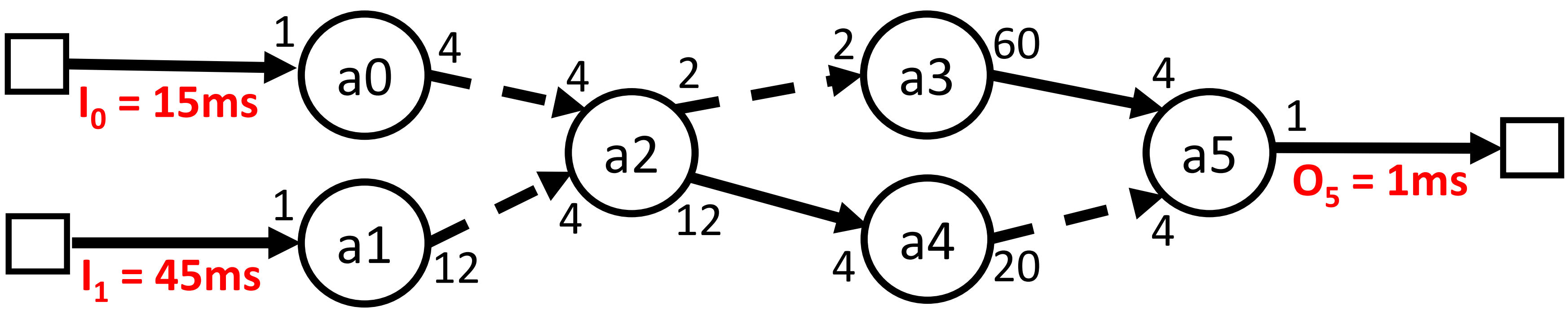
- **Model data losses in network channels as "empty" tokens**
  - Maintain deterministic execution in presence of losses
- **Channels could be lossless or lossy**
  - Traditional channels are "lossless"
  - "Lossy" channels can make a token "empty"
- **Actors need to handle network losses**
  - Can consume empty token(s), but produce only non-empty tokens
  - Multiple firing rules based on input token patterns



Timed Extension of RADF (T-RADF)

- **Empty tokens need to be injected by the runtime**
  - Decide based only on local time
  - ➢ Relative timeouts between firings

➢ **T-RADF extends RADF with rates on input and outputs**
  - Set (average) timeouts based on firing rates
  - Firing rates derived from external rates + repetition vector



Schedule Computation

- **Assumptions**
  - Homogeneous T-RADF
    - Any graph can be made homogeneous albeit exponentially larger
  - One actor per host
    - Statically schedule actors mapped to the same host into a super-actor
    - Might lead to deadlock, CSDF can relax this (future work)
- **Conservative analysis**
  - Fixed static schedule with specified periods
  - Can adjust schedule dynamically to optimize latency/quality
  - ➢ Derive relative start time offsets/phase shifts

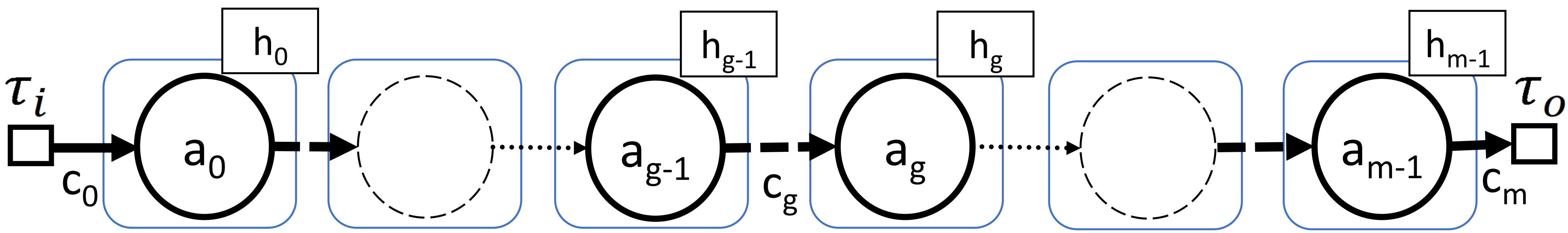
Schedule Computation

- **Latency l between input-output pair**
  - Depends on actor execution times e_i and channel delays d_j

$$l = \sum_{i=0}^{m-1} e_i + \sum_{j=1}^{m-1} d_j \leq l'$$

- **Latency constraint l' requires bounding e_i and d_j**
  - Worst-case execution time bounds: $e_i \leq e'_i$
  - Goal: find bounds d'_j for d_j

➢ **Find d'_j to satisfy l' and maximize output quality Q**
  - d'_j affects token delivery probability p_j and therefore quality
  - ➢ Quality model to describe Q in terms of d'_j



Quality Model (1)

- **SNR as quality metrics**
  - Signal processing applications
  - Quantify noise power of the output

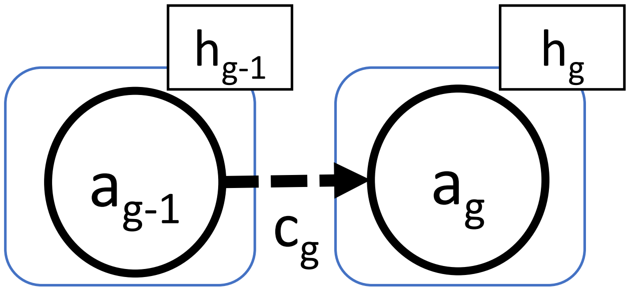
- **Single lossy channel**
  - Express token delivery probability p_g as function of delay budget d'_g

$$p_g(d'_g) = (1 - \mu_g)\,.\,F_{D_g}(d'_g)$$

with network parameters:

μ_g: average loss rate of channel c_g

$F_{D_g}(d_g)$: cumulative distribution function (CDF) of random network delay D_g



Quality Model (2)

- **Single lossy channel (cont'd)**
  - Noise in channel c_g at iteration i: $n_g[i] = \left|s_g[i] - R_g(s_g)\right|$

    $s_g[i]$: (original) signal value of c_g at iteration i

    R_g: replacement function of actor a_g

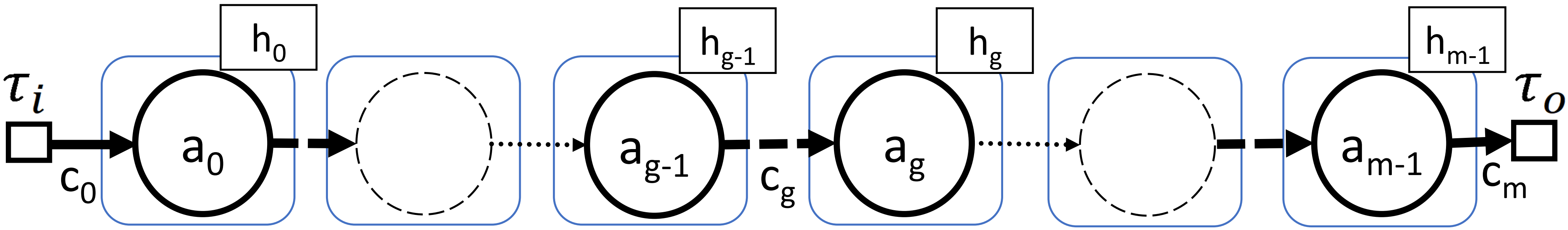
  - Assuming n_g is upper bounded by s_g: $n_g[i] \leq \alpha s_g[i]$

- **Actor chain**
  - Assuming system and, hence, actors are linear with weights w_j

    $$n_m[i] \leq \alpha \widetilde{w} s_0[i] \qquad \text{where } \widetilde{w} = \prod_{j=1}^{m} w_j$$

    – This equation holds regardless of number of losses

➢ **Calculate minimum SNR from maximum noise power**



Quality Model Computation

- **In graph with more than one actor chain**
  - Iterate over paths and accumulate noise power
  - Multiple iterations for cyclic graphs
  - Exponential complexity
- **A better alternative**
  - Breadth-first traversal of channels
  - Accumulate partial noise power at each channel
  - $O\left(I * m^2/n\right)$ complexity
    - in number of inputs I, channels m and nodes n

Scheduling Formulation

- **Optimization problem: find the schedule that maximizes quality**

$$\underset{\mathbf{d}'}{\text{maximize}} \quad Q(\mathbf{d}', \tau)$$

$$\text{subject to} \quad d'_j \geq 0,\ \forall j \in channels. \qquad \text{precedence constraints}$$

$$\sum_{i=0}^{m_k-1} e'_{(k,i)} + \sum_{j=1}^{m_k-1} d'_{(k,i)} \leq l'_k,\ \forall k \in paths. \qquad \text{latency constraints}$$

- **Solving the optimization problem**
  - $\boldsymbol{Q}$ is non-linear/non-convex but has closed form and is differentiable
  - $\boldsymbol{d}'$ are continuous
  - ➢ Use numerical gradient-based iterative methods
    - ➢ E.g. Constrained Trust Region (CTR) solver

Outline

- ✓ **Introduction**
  - ✓ Motivation, background
  - ✓ Related work
- ✓ **Formalizing distributed data flow**
  - ✓ Timed extension of RADF
- ✓ **Scheduling distributed data flow**
  - ✓ Quality model and optimization
- **Experimental Results**
- **Summary & Future Work**

Experimental Setup (1)

- **T-RADF and scheduler implemented in Python**
  - NetworkX for graph representation
  - Scipy for optimization with 10^{-5} as gradient norm threshold
  - 5 iterations for cyclic graphs

- **Application models**
  - 100 random cyclic/acyclic graphs with 10/50/100 nodes [sdf3]
  - Three replacement policies
    - R_{static}: replaces empty tokens with zeros
    - R_{last}: replaces empty tokens with the last received value
    - R_{avg}: replaces with the running average of received values

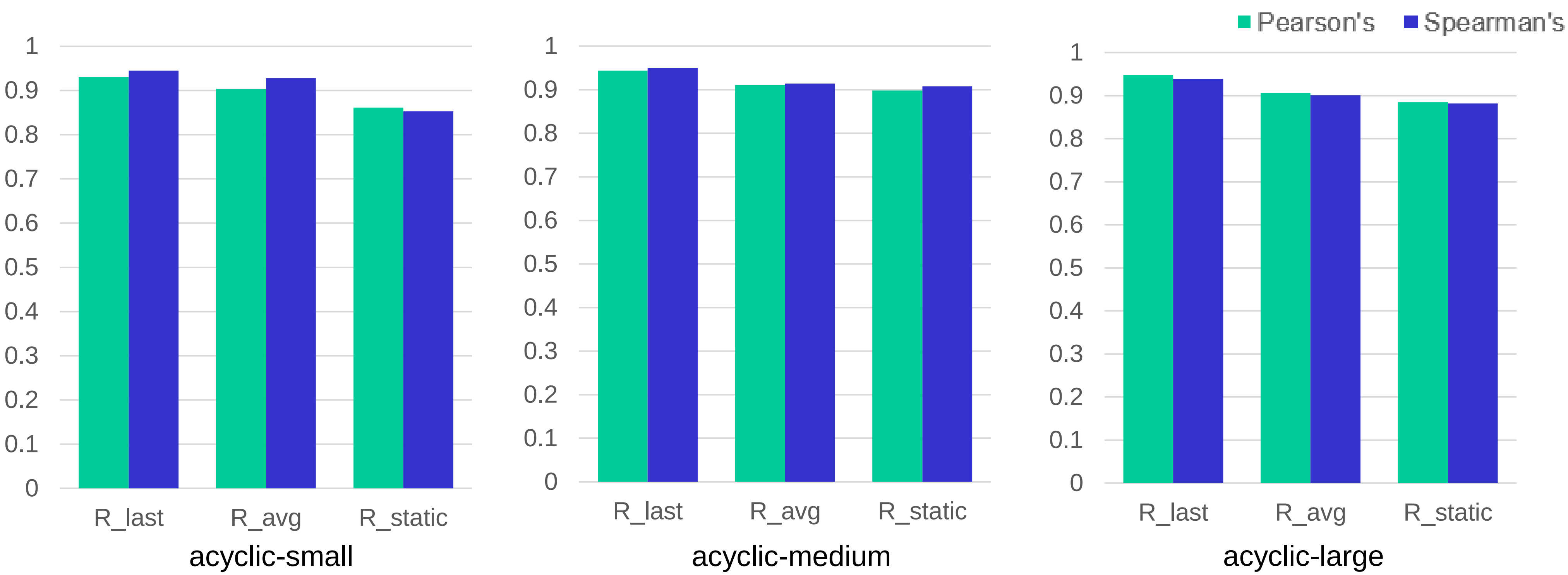
Experimental Setup (2)

- **Simulated via OMNET++ and INET Framework**
  - UDP sockets for lossy channels
  - INET's cloud model (5Mbps, μ = 1%, Gamma NDD)

- **Relative latency constraints**
  - l_{min}/l_{max} are latencies for delivery probabilities of 0.1% and 99.9%
  - Constraint factor $\rho = (l' - l_{min})/(l_{max} - l_{min})$

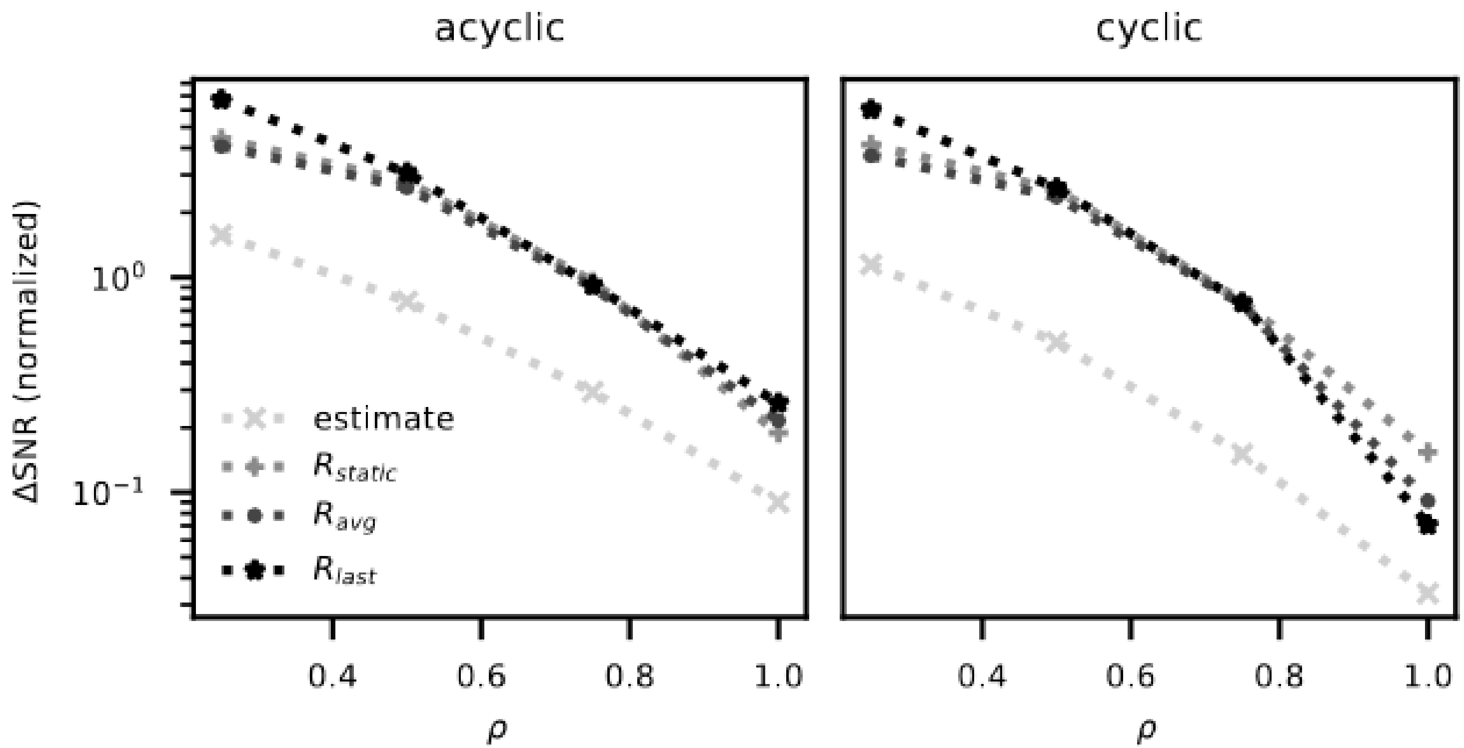
Quality Model Fidelity

- **Correlation between estimated and measured SNR**
  - Relatively/monotonicity: Spearman's correlation coefficient
  - Absolute values: Pearson's correlation coefficient
- **Setup**
  - Generated 100 random schedules for ten graphs
  - Latency constraint factor ρ randomly chosen in the interval [0.1,0.9]



Optimization Results

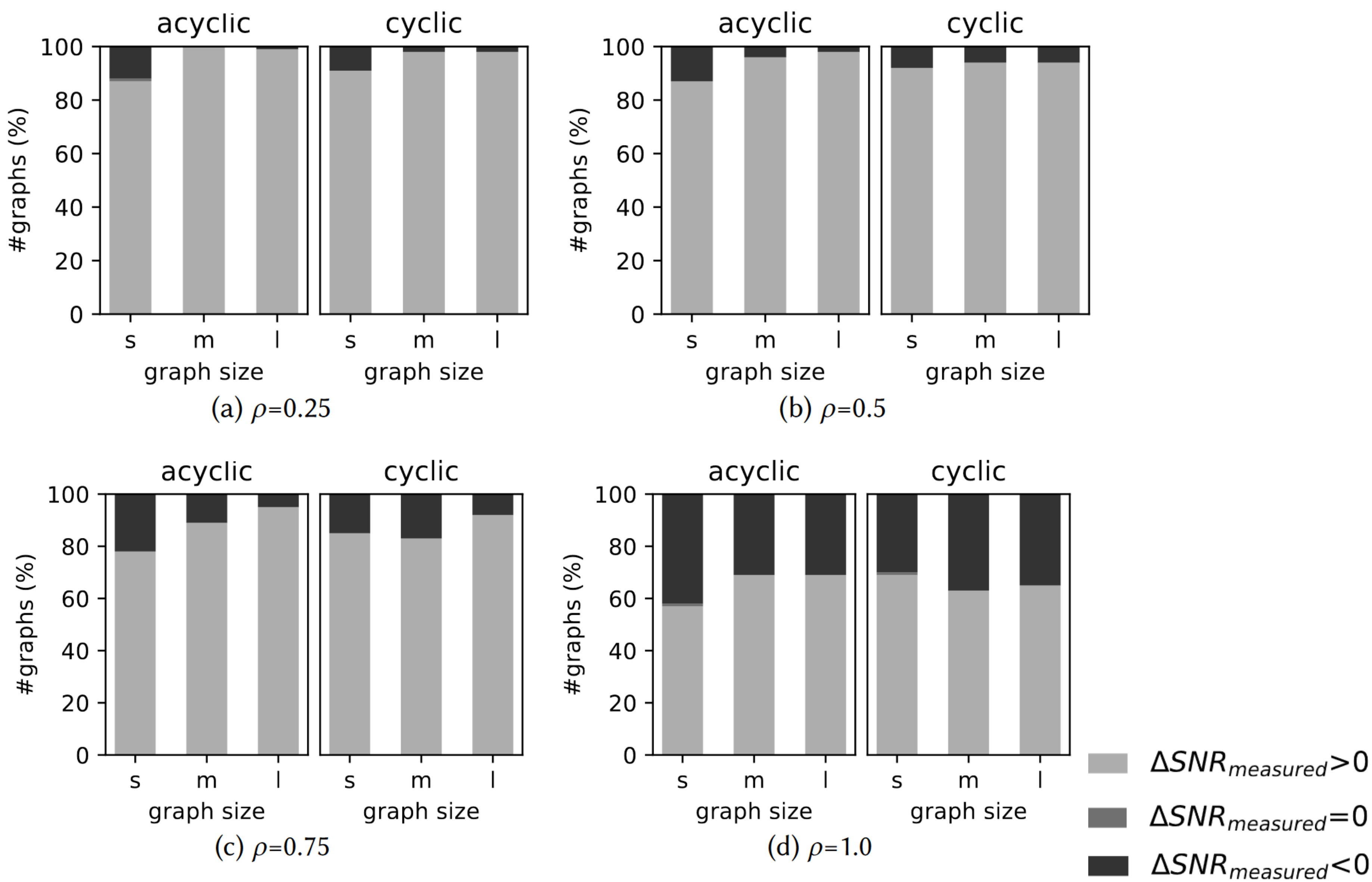
- **Measured and predicted SNR improvement**
  - Vs. uniform budget distribution as baseline schedule
  - Averaged across sizes



- ➢ **Higher gains with tight constraints**
  - ➢ Up to 900%, on average 75%
- ➢ **Conservative estimate from quality model**

Optimization Outcomes

- **Optimization success rate**



(a) ρ=0.25

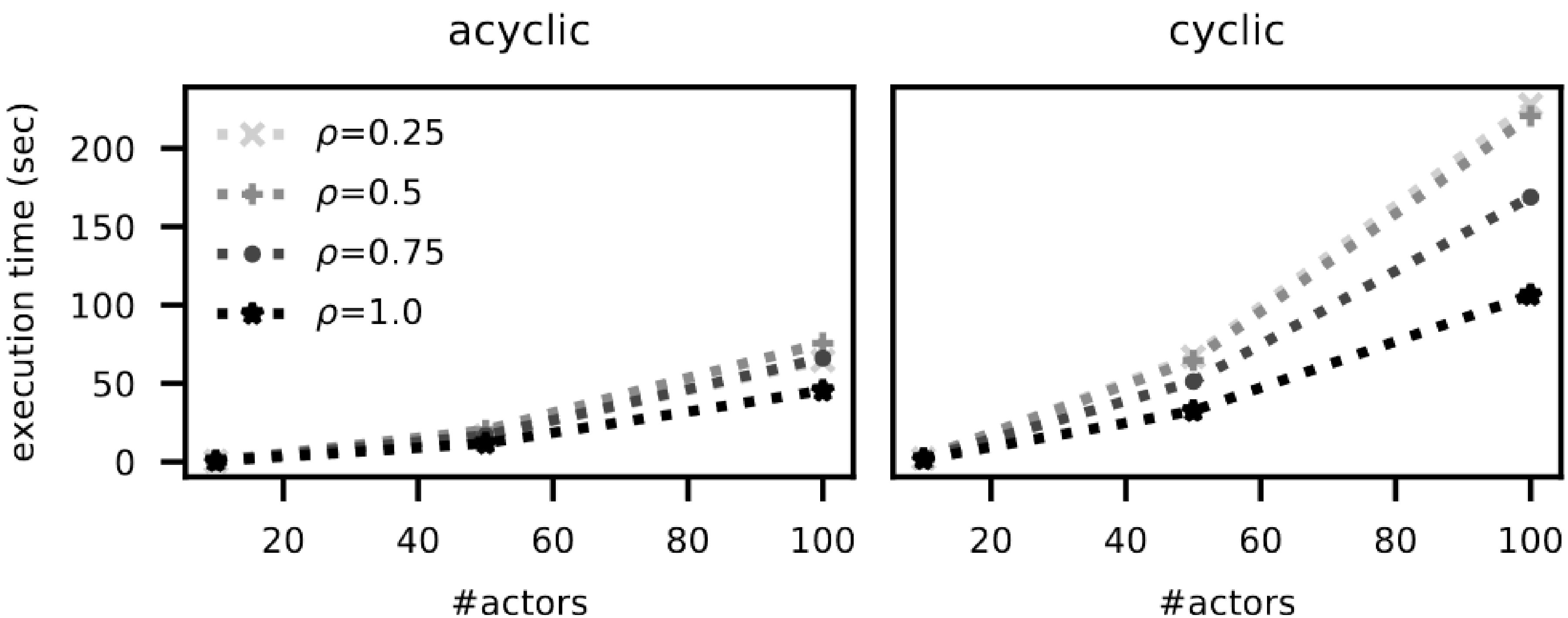
(b) ρ=0.5

(c) ρ=0.75

(d) ρ=1.0

Optimization Runtime

- **Average runtime of optimization solver**
  - Measured on Intel Core i7-920

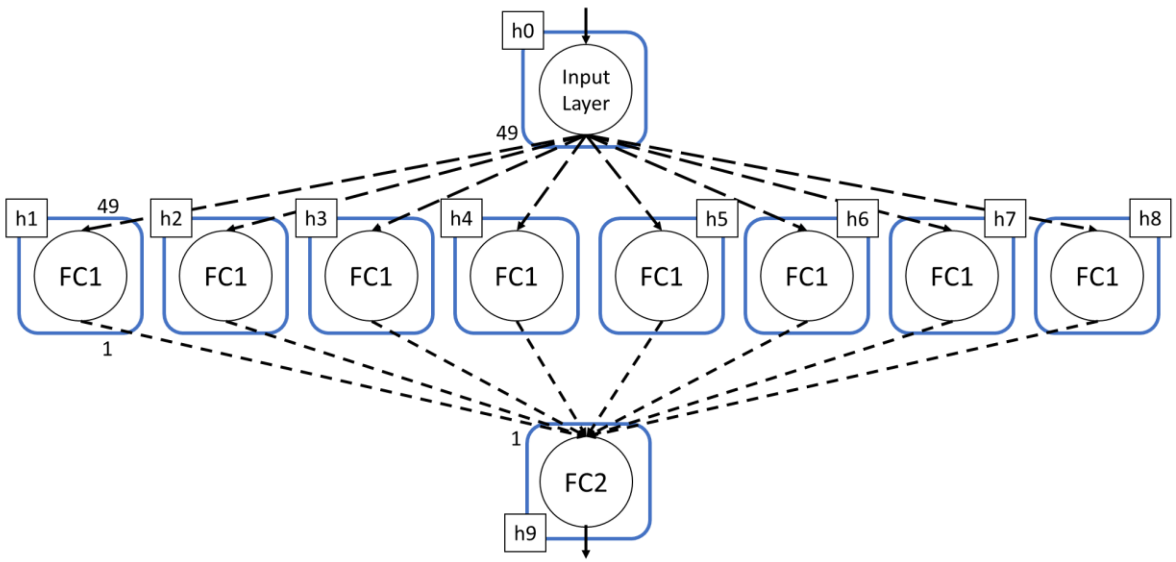


➢ **Larger for tight constraints**

➢ **Larger for cyclic graphs due to multiple iterations**

Distributed Neural Network Example

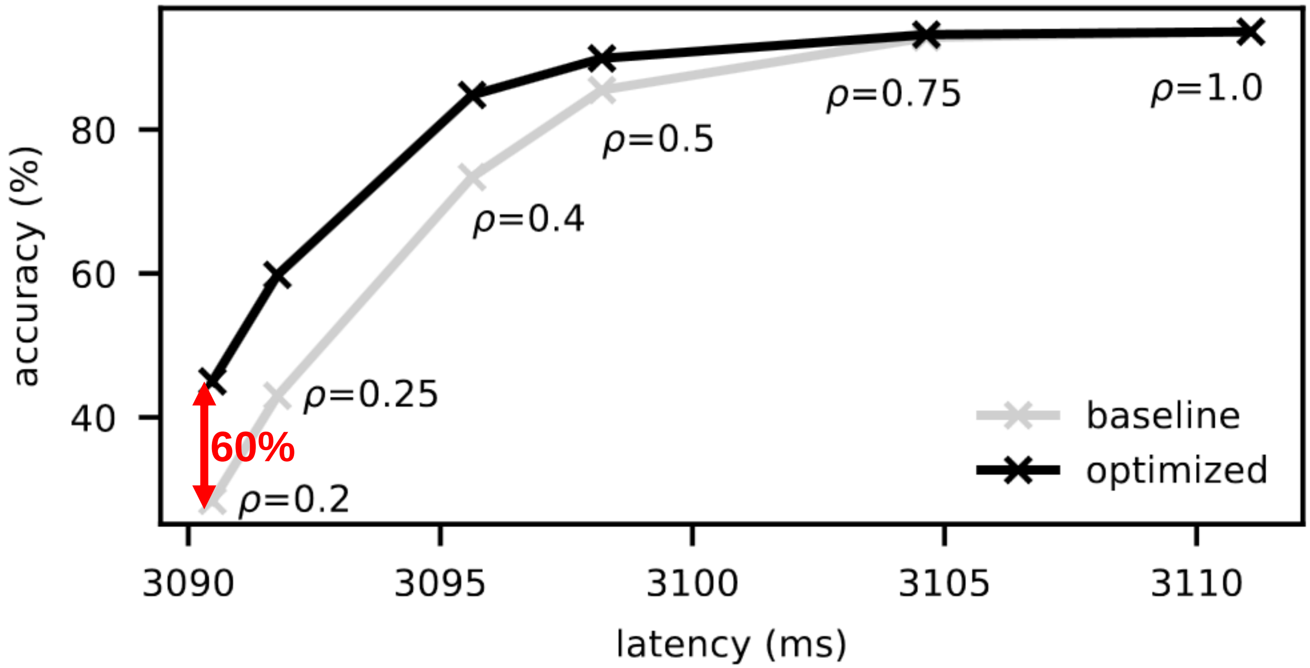
- **Two-layer network for MNIST digit recognition**
  - Non-linear due to activation functions
  - Each token is 16 doubles
  - WCET of 1sec for FC1/FC2
  - Account for network delay of 49 tokens in WCET of Input



- **Significant accuracy gains under tight latency constraints**
  - Up to 60%, on average 20%

Distributed Neural Network Example

- **Two-layer network for MNIST digit recognition**
  - Non-linear due to activation functions
  - Each token is 16 doubles
  - WCET of 1sec for FC1/FC2
  - Account for network delay of 49 tokens in WCET of Input



➢ **Significant accuracy gains under tight latency constraints**
  - ➢ Up to 60%, on average 20%

Summary and Future Work

- **Quality/latency-aware scheduling**
  - Dataflow models for distributed embedded systems
  - Quality/latency tradeoff
  - Scheduling dist. dataflow by optimizing the tradeoff
  - ➢ Tools, graphs and simulation models available online

    https://github.com/SLAM-Lab/QLA-RTS

- **Future work**
  - Address the restrictions
    - Homogeneity, one-actor-per-host mapping
  - Runtime system for T-RADF
    - Dynamic scheduling