UNIVERSITÄT BERN

Seminar Communication and Distributed Systems

Service Distribution Mechanisms in Information-Centric Networking

Bachelor Thesis of

Oliver Stapleton

11.05.2015

Outline

- Introduction
- Keywords
- Approaches
  - Trivial Approach
  - Probability-based Approach
- Evaluation
- Conclusion
- Future Work

Thesis

- Topic:
  - “Service Distribution Mechanisms in Information-Centric Networking“

- Motivation:
  - Current concept of ICN allows objects to be distributed, but the processing of services relies on a single ad hoc server
  - Distribution of a task by splitting it into several parts and distributing them to separate nodes has not been implemented yet

Thesis

> Tasks at a glance:

— get familiar with ICN / CCN / SCN

— design a service distribution mechanism

— implement it

— evaluate its performance

UNIVERSITÄT BERN

Keywords

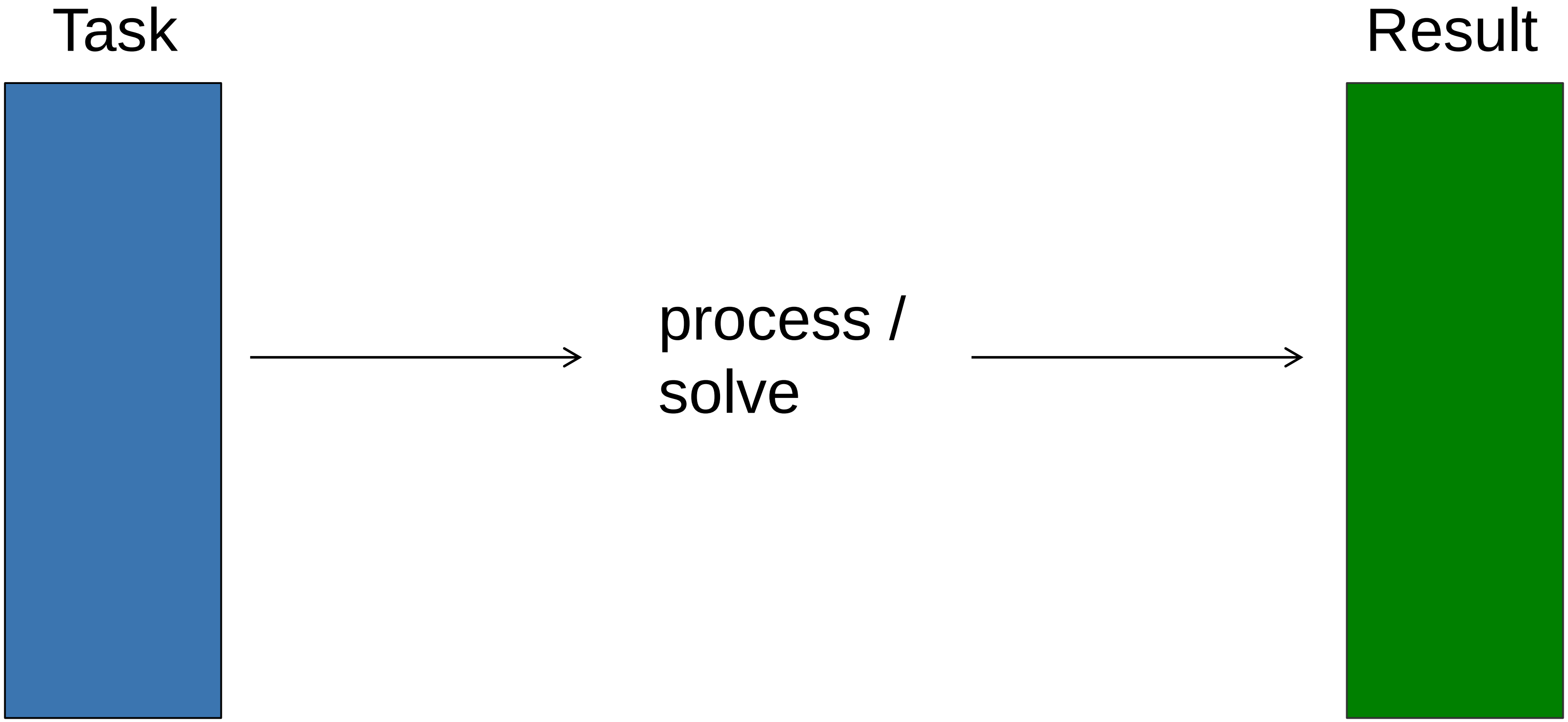
- ICN: Information-Centric Network
  - decoupling of information and location
  - Consumers only express name of information, not location
  - Routing takes care of retrieving information from closest source

  - 2 kinds of messages:
    - Interest Message (Request): „/service/factorial/!{N=“5"}“
    - Data Message (Information): „120“

- CCN: ICN, where information = (static) content
- SCN: ICN, where information = service

- CCNx: implementation of CCN
- NextServe: extension of CCNx that adds SCN-capability

Task distribution

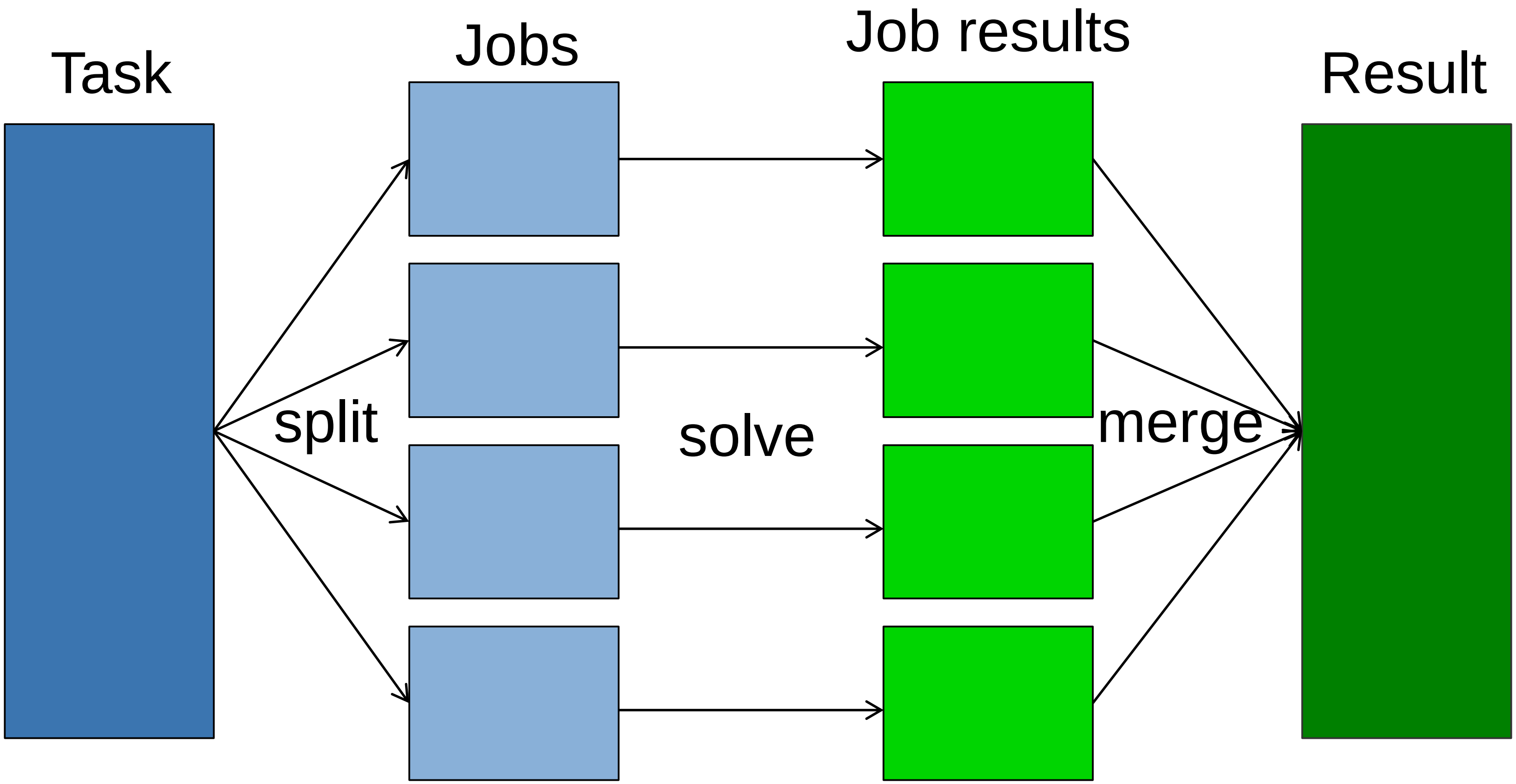


Task

process / solve

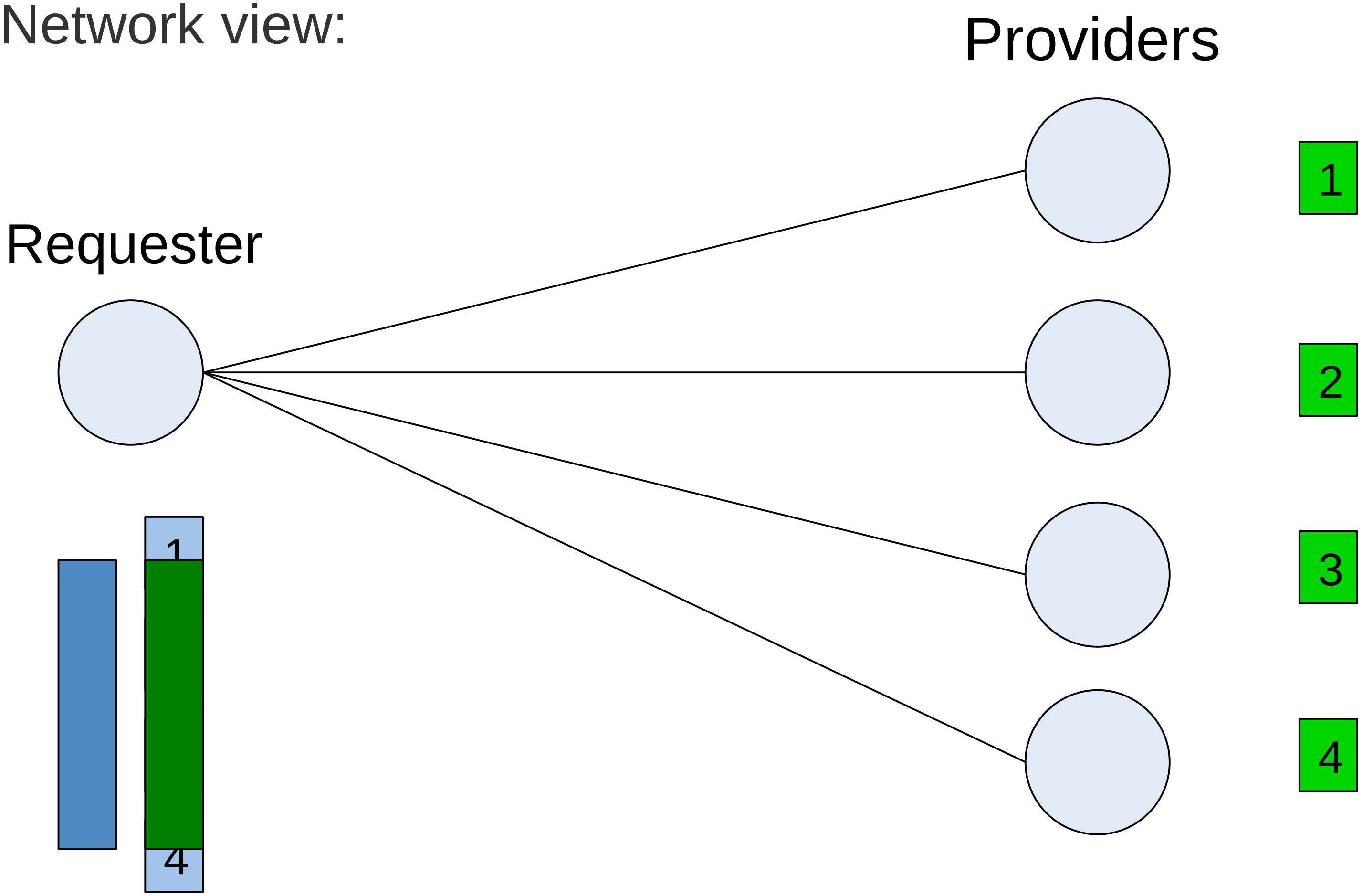
Result

Task distribution



Task distribution

Network view:



Goals of Task distribution

- Time
  - Minimize turnaround time for Tasks

- Optimize power consumption
  - Exploit the available resources of other, idle nodes in the network
  - Minimize power consumption of Requester
    - Requester might be mobile/running on a limited battery

Considered approaches

1. Trivial Approach

~~2. Delegated Approach~~

3. Probability-based Approach

Trivial Approach (TA)

1. Requester defines Task and splits it into *n* Jobs

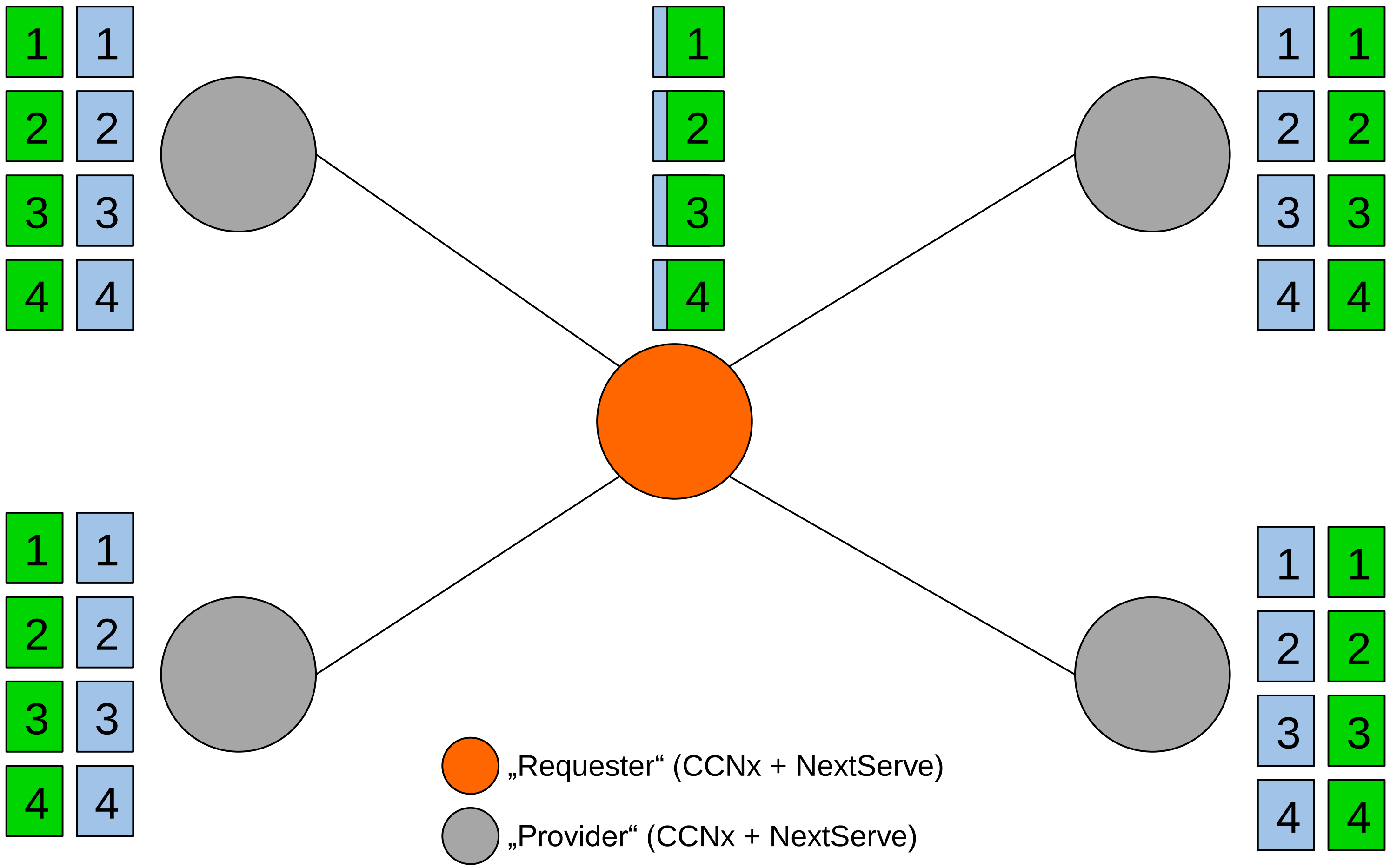
2. Requester expresses each Job as an Interest Message (IM)

3. Providers receive IMs (aka the Jobs)

4. Providers process IMs, and return the Job results as Data Messages (DMs)

5. Requester waits until all *n* DMs (aka Job results) have arrived and then merges them to form the overall Task result

Trivial Approach (TA)



● „Requester“ (CCNx + NextServe)

● „Provider“ (CCNx + NextServe)

Trivial Approach (TA)

- Problems with the trivial approach:
  - IMs aren‘t „delegated“ to separate Providers
    - Each Provider receives each IM
    - Each Provider processes each IM
    - Requester must wait for fastest Provider to have processed all IMs
  - No gain over just sending out the whole Task as a single IM!

Probability-based Approach (PBA)

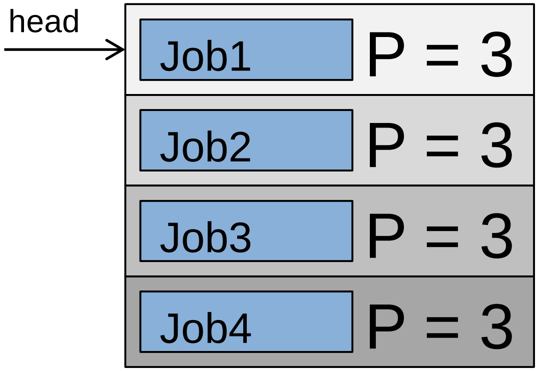
- Mechanism to solve the issues of the Trivial Approach

- Order of incoming IMs is „shuffled“ by the Providers

- Probability-based:
  - Providers queue incoming IMs
  - IMs contain P-value
  - Providers iterate through queue
  - For each IM, they „throw a dice“:
    - In 1 out of P times, they decide to process the IM
    - In P-1 out of P times, they add the IM back to the queue

Probability-based Approach (PBA)

Queue of incoming IMs:



head

Job1 P = 3

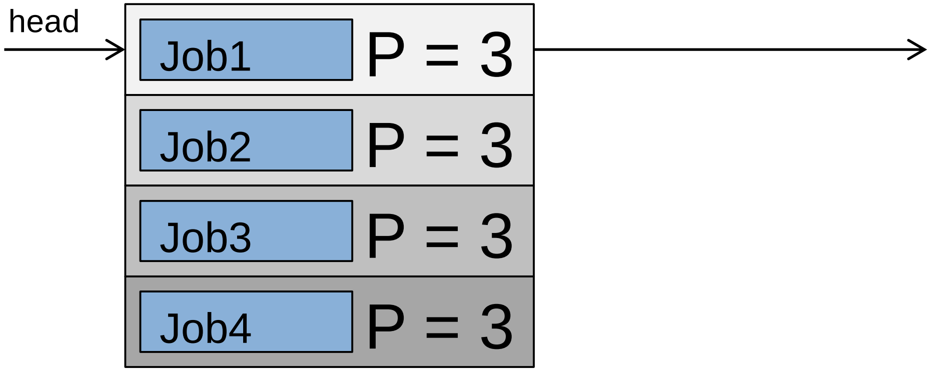
Job2 P = 3

Job3 P = 3

Job4 P = 3

Probability-based Approach (PBA)

Queue of incoming IMs:



head

Job1 P = 3

Job2 P = 3

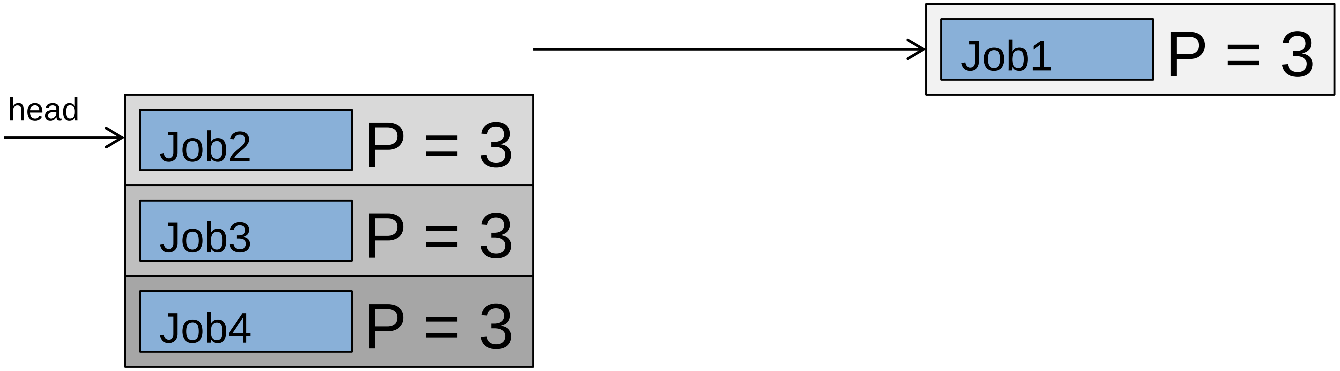
Job3 P = 3

Job4 P = 3

1. Dequeue top IM

Probability-based Approach (PBA)

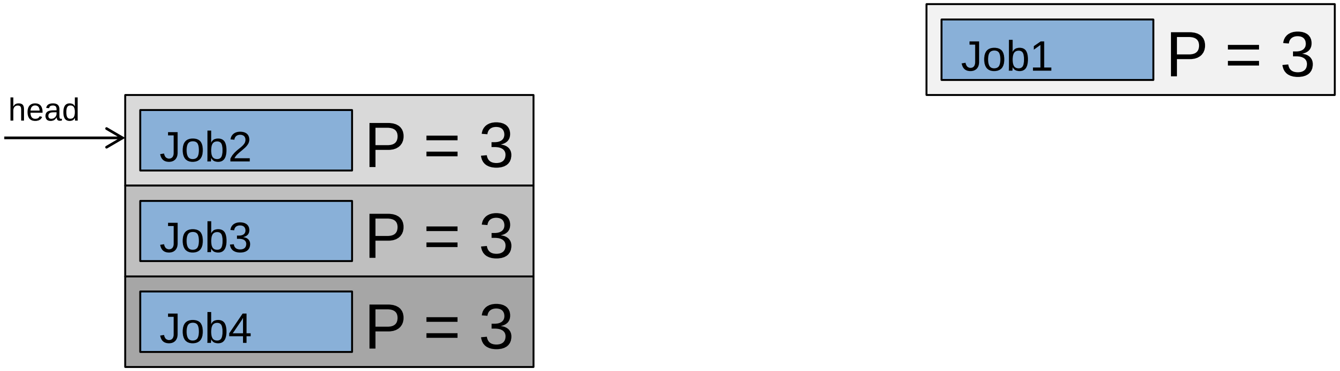
Queue of incoming IMs:



1. Dequeue top IM

Probability-based Approach (PBA)

Queue of incoming IMs:



Job1 P = 3

head

Job2 P = 3

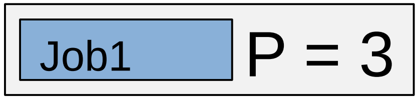
Job3 P = 3

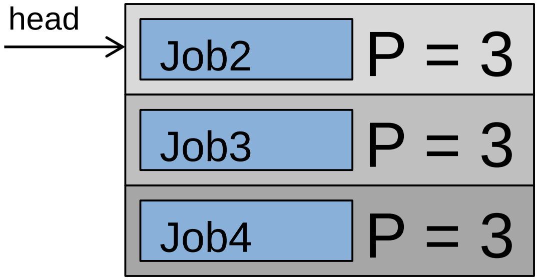
Job4 P = 3

2. Read P-value

Probability-based Approach (PBA)

Queue of incoming IMs:


Job1 P = 3


head

Job2 P = 3

Job3 P = 3

Job4 P = 3

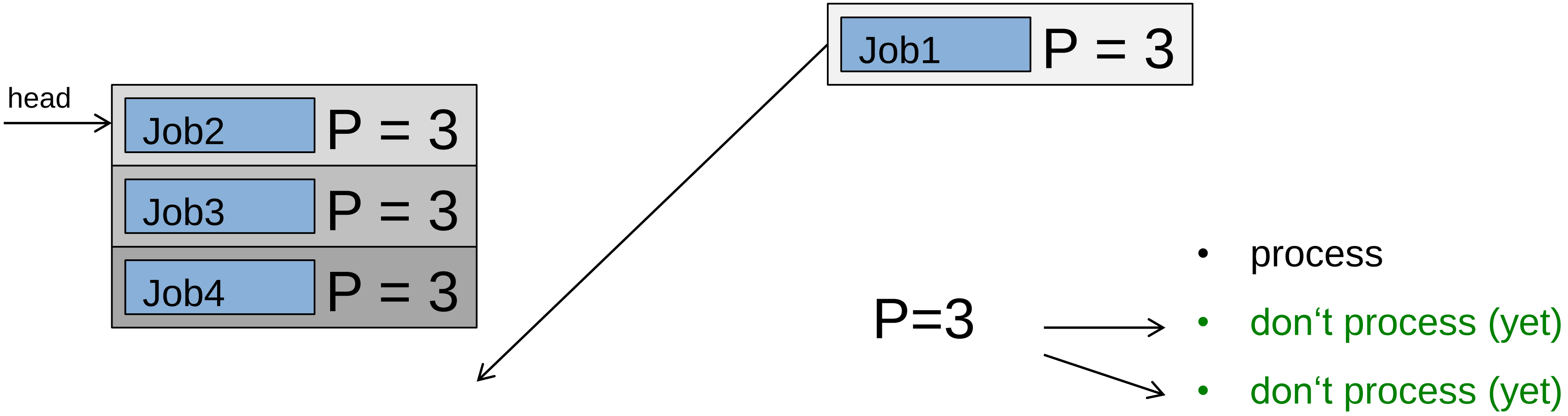
P=3

- process
- don‘t process (yet)
- don‘t process (yet)

2. Read P-value

Probability-based Approach (PBA)

Queue of incoming IMs:

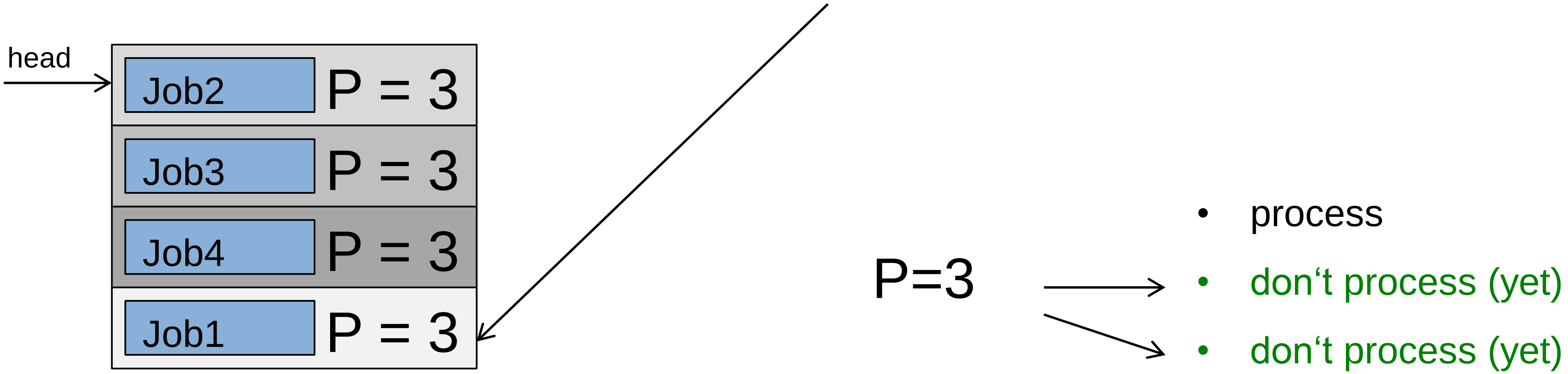


- process
- don‘t process (yet)
- don‘t process (yet)

3. Decision (B & C): Enqueue IM again

Probability-based Approach (PBA)

Queue of incoming IMs:



head

Job2	P = 3
Job3	P = 3
Job4	P = 3
Job1	P = 3

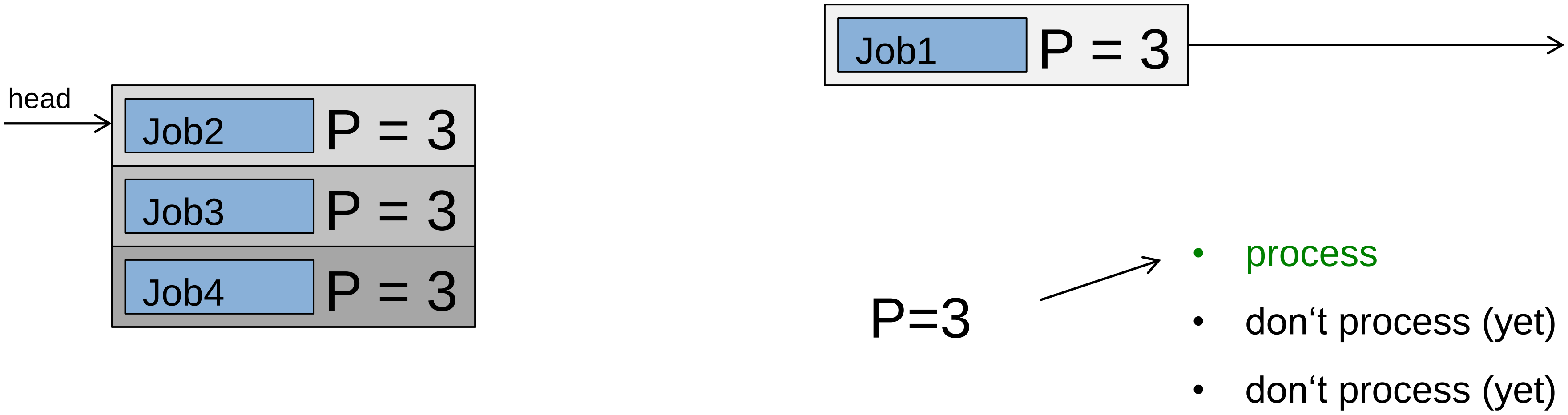
P=3

- process
- don‘t process (yet)
- don‘t process (yet)

3. Decision (B & C): Enqueue IM again

Probability-based Approach (PBA)

Queue of incoming IMs:

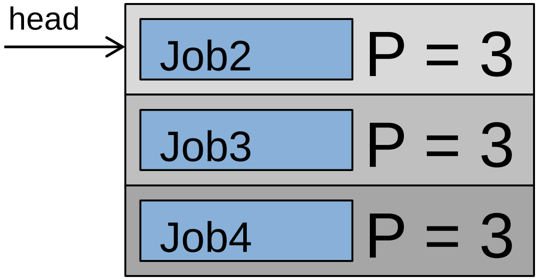


- process
- don‘t process (yet)
- don‘t process (yet)

3. Decision (A): Process the Job within the IM

Probability-based Approach (PBA)

Queue of incoming IMs:



head

Job2 P = 3

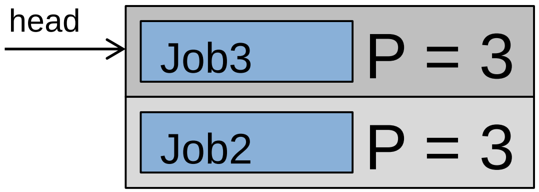
Job3 P = 3

Job4 P = 3

4. Repeat until no IMs left in queue

Probability-based Approach (PBA)

Queue of incoming IMs:



head

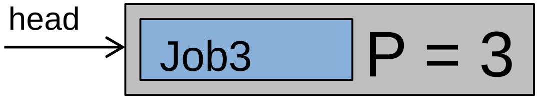
Job3 P = 3

Job2 P = 3

4. Repeat until no IMs left in queue

Probability-based Approach (PBA)

Queue of incoming IMs:



4. Repeat until no IMs left in queue

Probability-based Approach (PBA)

Queue of incoming IMs:

head →

4. Repeat until no IMs left in queue

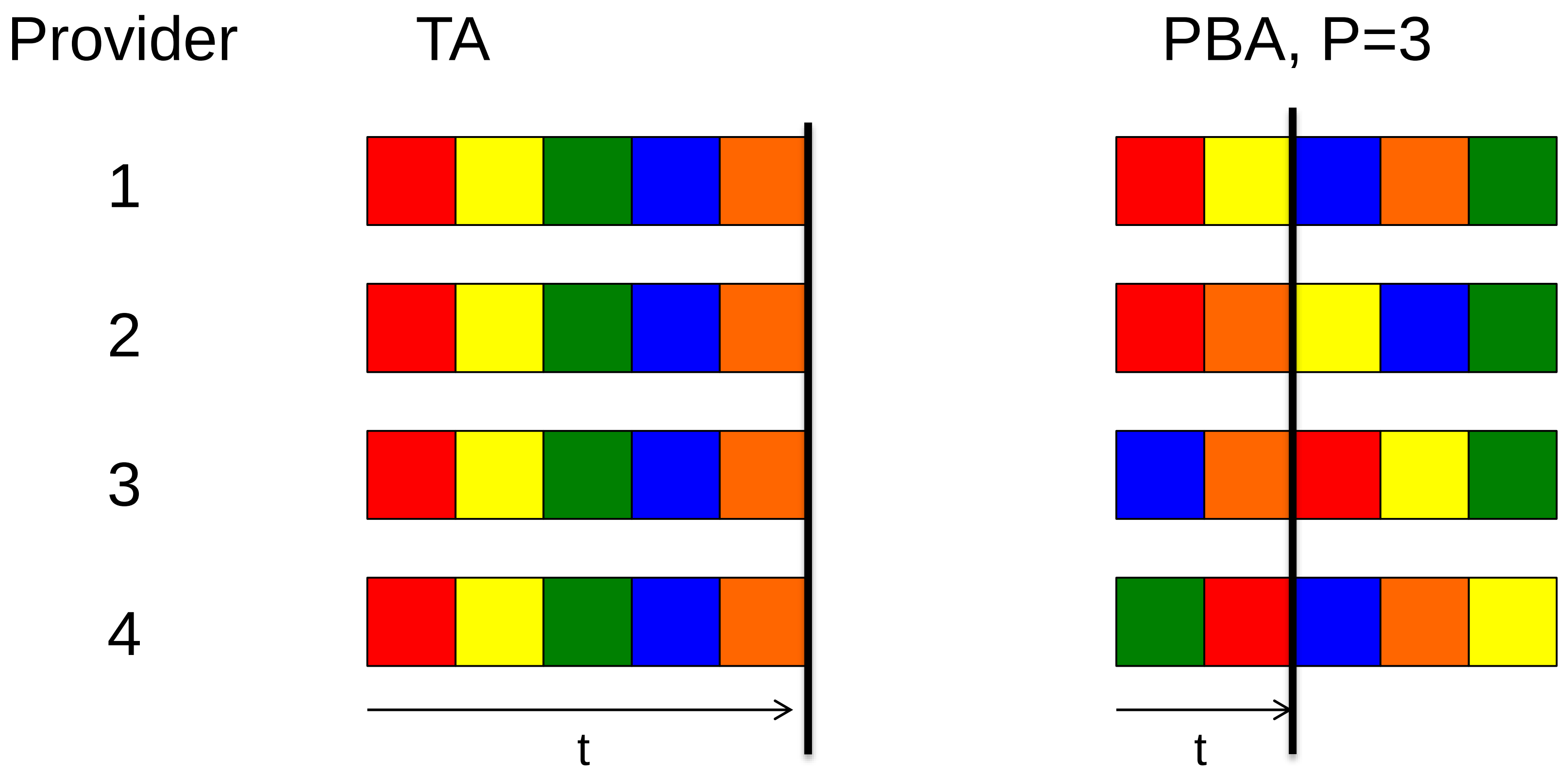
Probability-based Approach

> Simulation: Order that Providers process incoming IMs

Provider	TA	PBA, P=3
1	🟥🟨🟩🟦🟧	🟥🟨🟦🟧🟩
2	🟥🟨🟩🟦🟧	🟥🟧🟨🟦🟩
3	🟥🟨🟩🟦🟧	🟦🟧🟥🟨🟩
4	🟥🟨🟩🟦🟧	🟩🟥🟦🟧🟨

Probability-based Approach

- Simulation: Order that Providers process incoming IMs



Evaluation

- Only PBA
  - TA is actually special case of PBA with P=1

- Two phases
  - Phase 1: Evaluation of PBA on model
    - Model simulates ICN with PBA distribution mechanism
    - To prove that PBA does actually reduce turnaround-time
    - Find out what is the best value for P, amount of Jobs

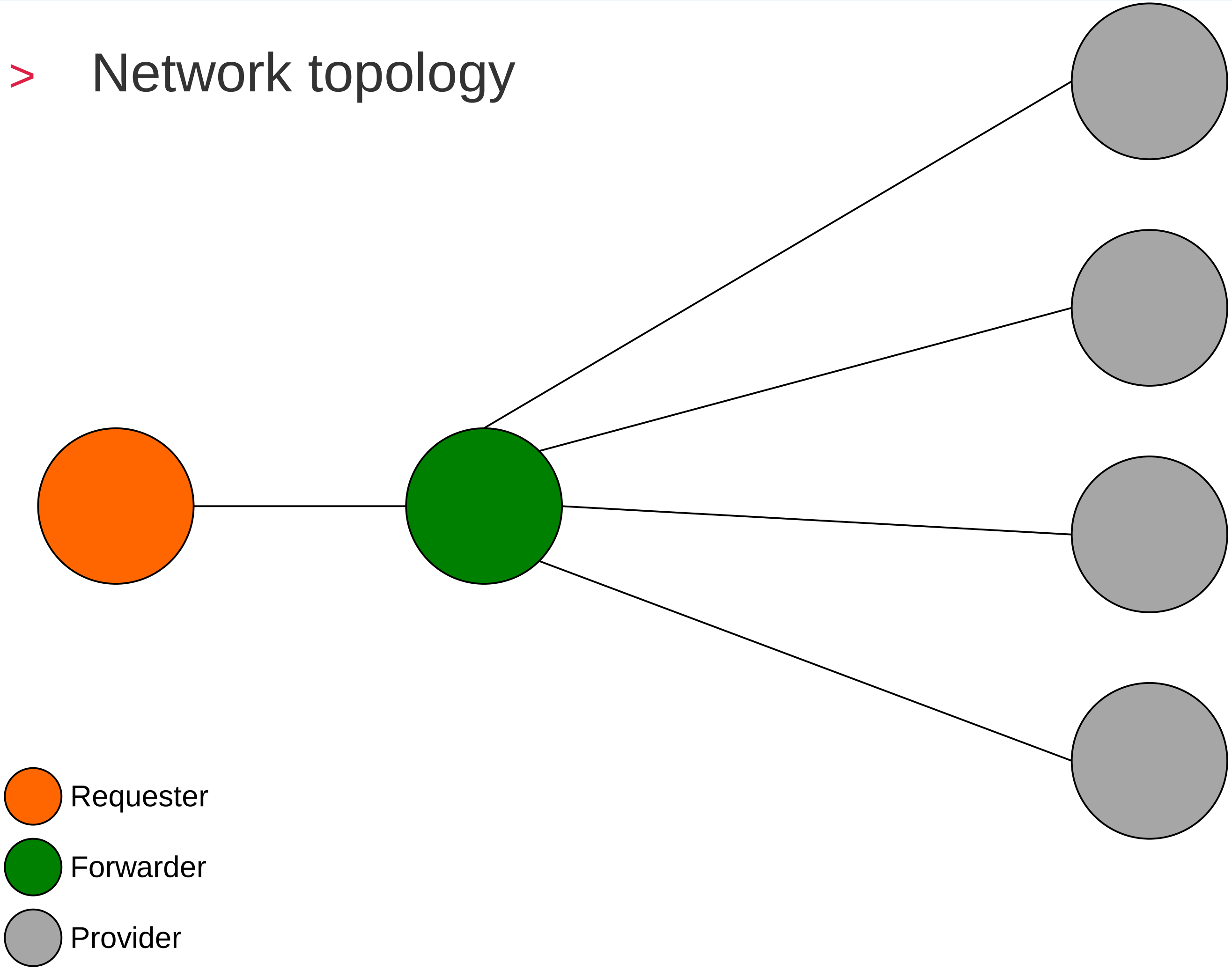
  - Phase 2: Show that PBA also works when implemented in CCNx/NextServe

Evaluation

- Task: Count from 1 to 100, one number per second
  - Takes 100s to solve locally
  - Can be split into independent Jobs
    - 2: [count: 1-50],[count: 51-100]
    - 4: [count: 1-25], [count: 26-50], [count: 51-75], [count: 76-100]
    - etc.
  - Dummy Task to minimize outside influences on measurement
    - no data transfer
    - multiple Providers can be powered by same CPU

Evaluation

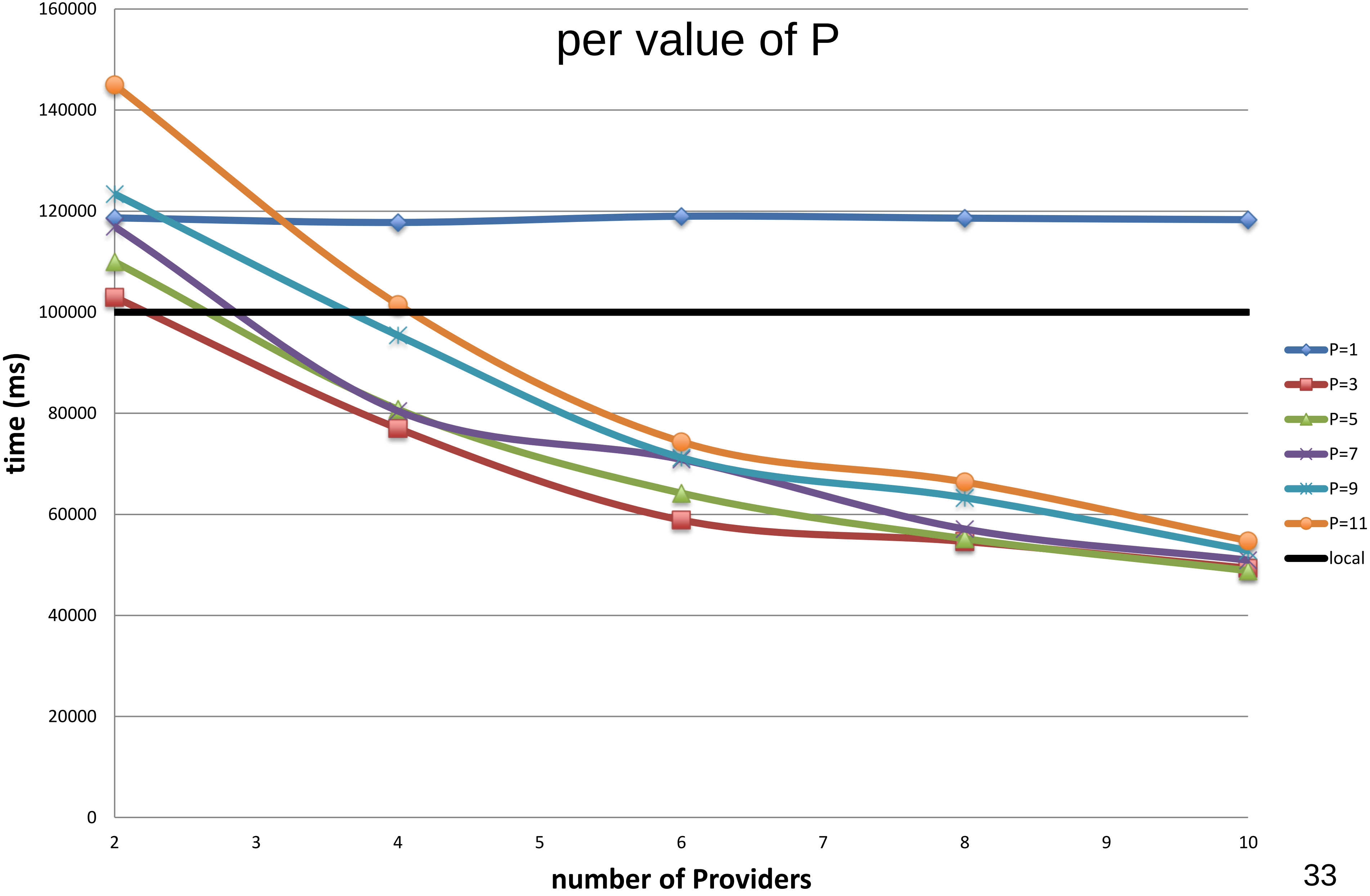
> Network topology



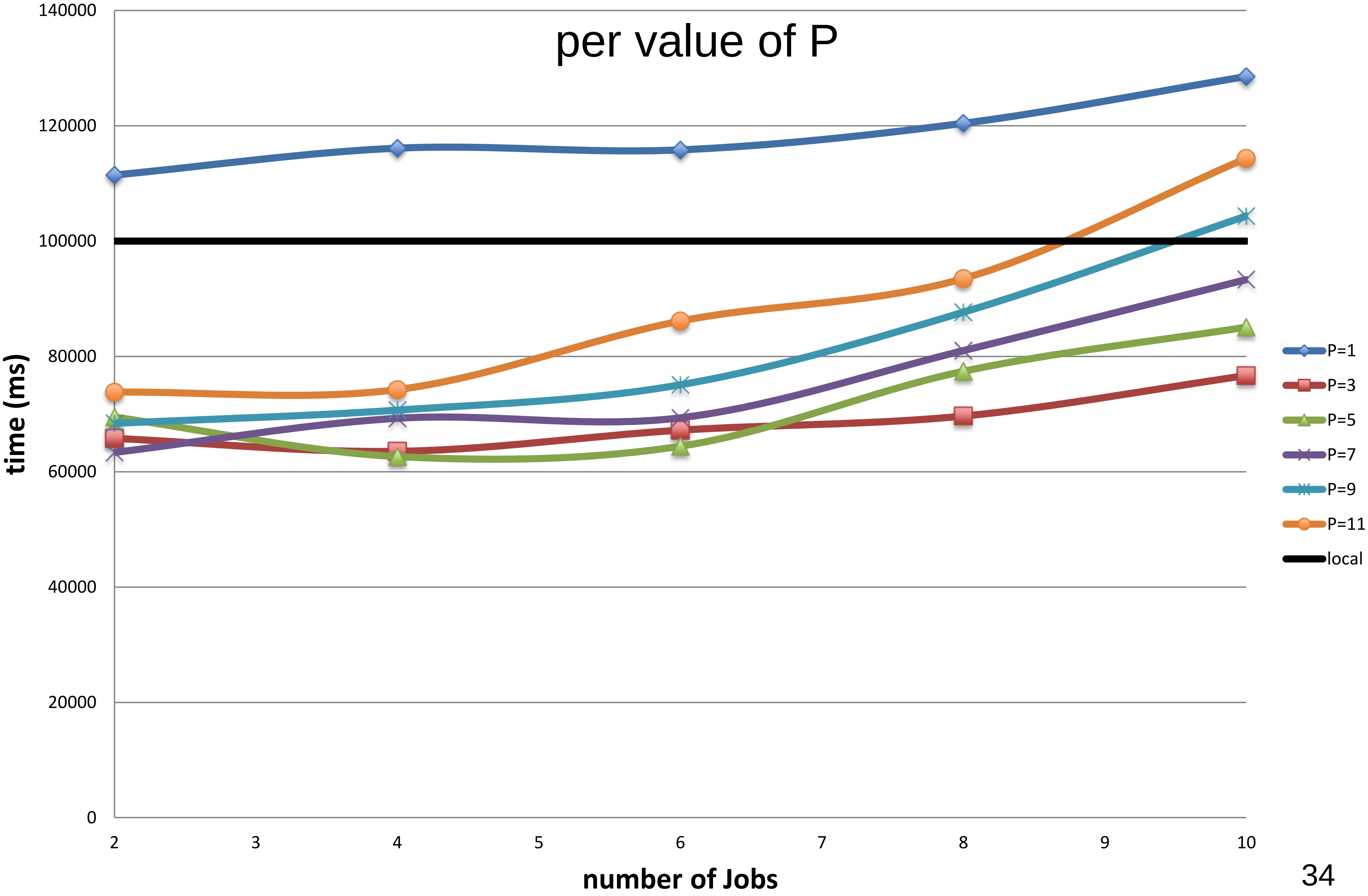
Evaluation

- Phase 1, session 1:
- Varying inputs for
  - Number of Jobs => {2, 4, 6, 8, 10)
  - Number of Providers => {2, 4, 6, 8, 10}
  - Value for P => {2, 4, 6, 8, 10, 12}
- 10 measurements for each combination of inputs

Mean time to solve task depending on amount of **Providers**, per value of P



Mean time to solve task depending on amount of **Jobs**, per value of P



Evaluation

- Phase 1, session 1:

- Conclusions:

  - PBA minimizes turnaround-time for task in most cases

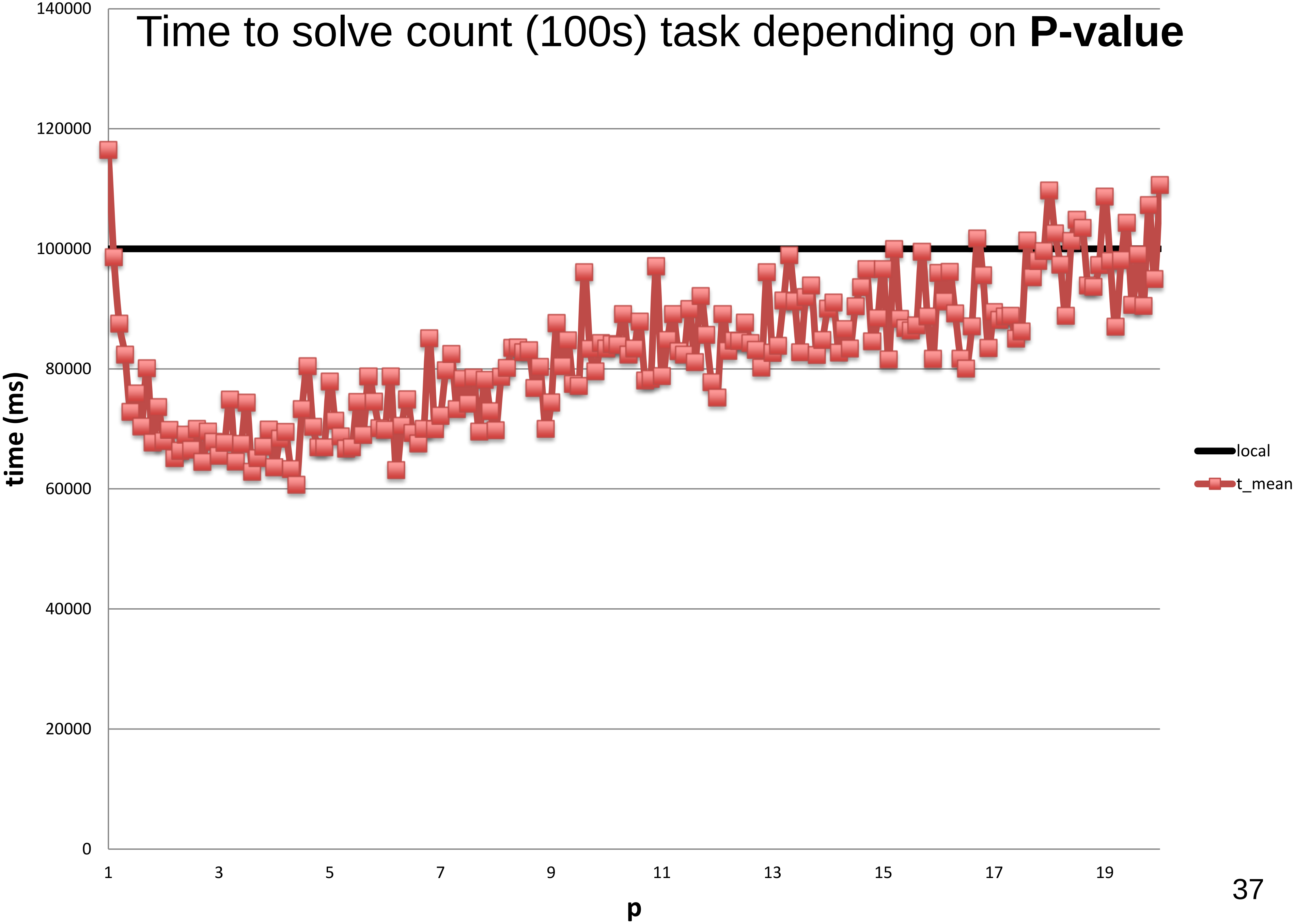
  - The more Providers, the better the performance (obviously)

  - Best performance for varying numbers of providers is achieved with:

    Number of jobs = 4

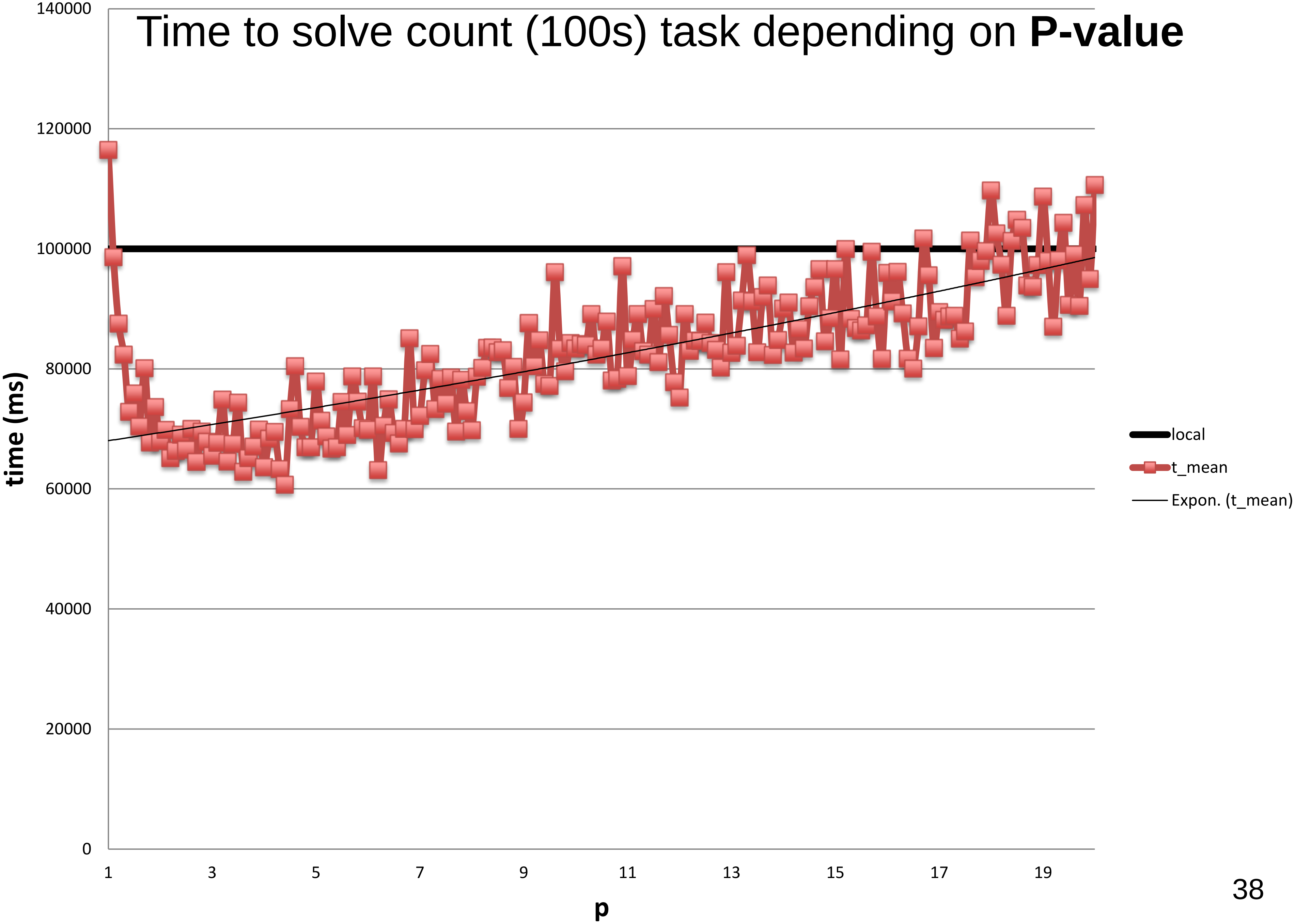
Evaluation

- Phase 1, session 2:
- Constant inputs for
  - Number of Jobs => 4
  - Number of Providers => 4
- Varying inputs for
  - Value for P => {1, 1.1, 1.2, ..., 19.8, 19.9, 20}
- 20 measurements for each set of inputs

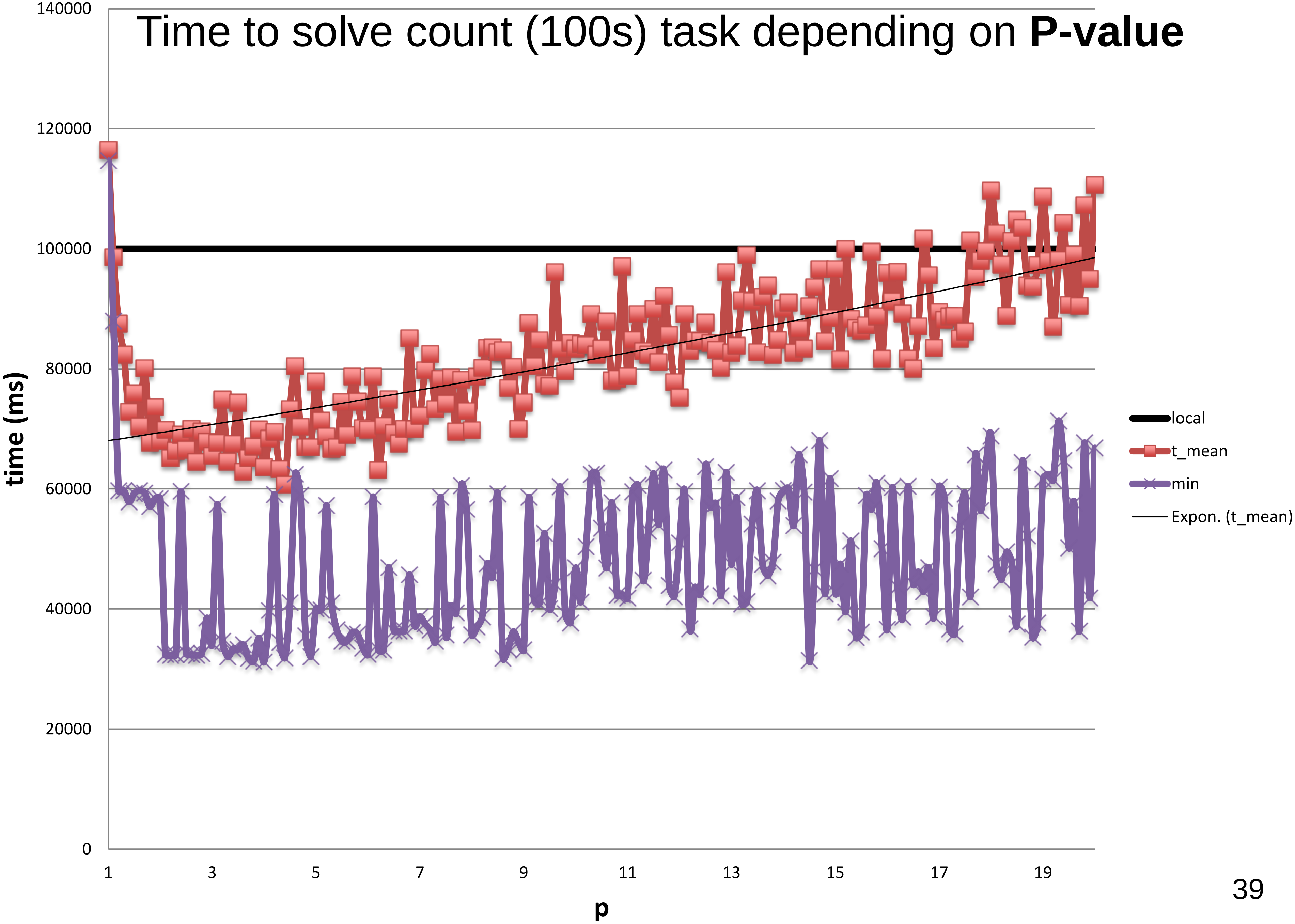


Time to solve count (100s) task depending on **P-value**

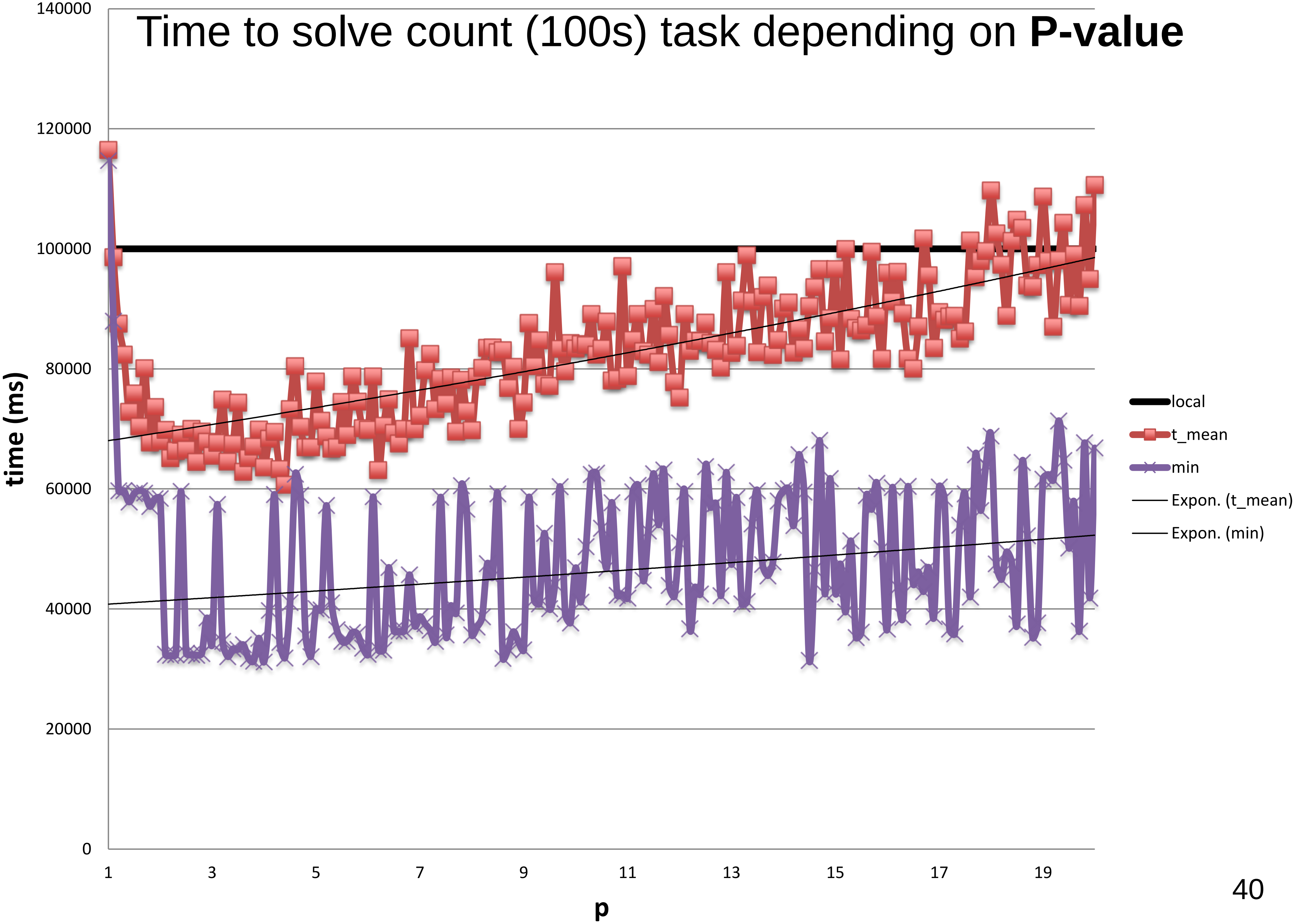
Time to solve count (100s) task depending on **P-value**

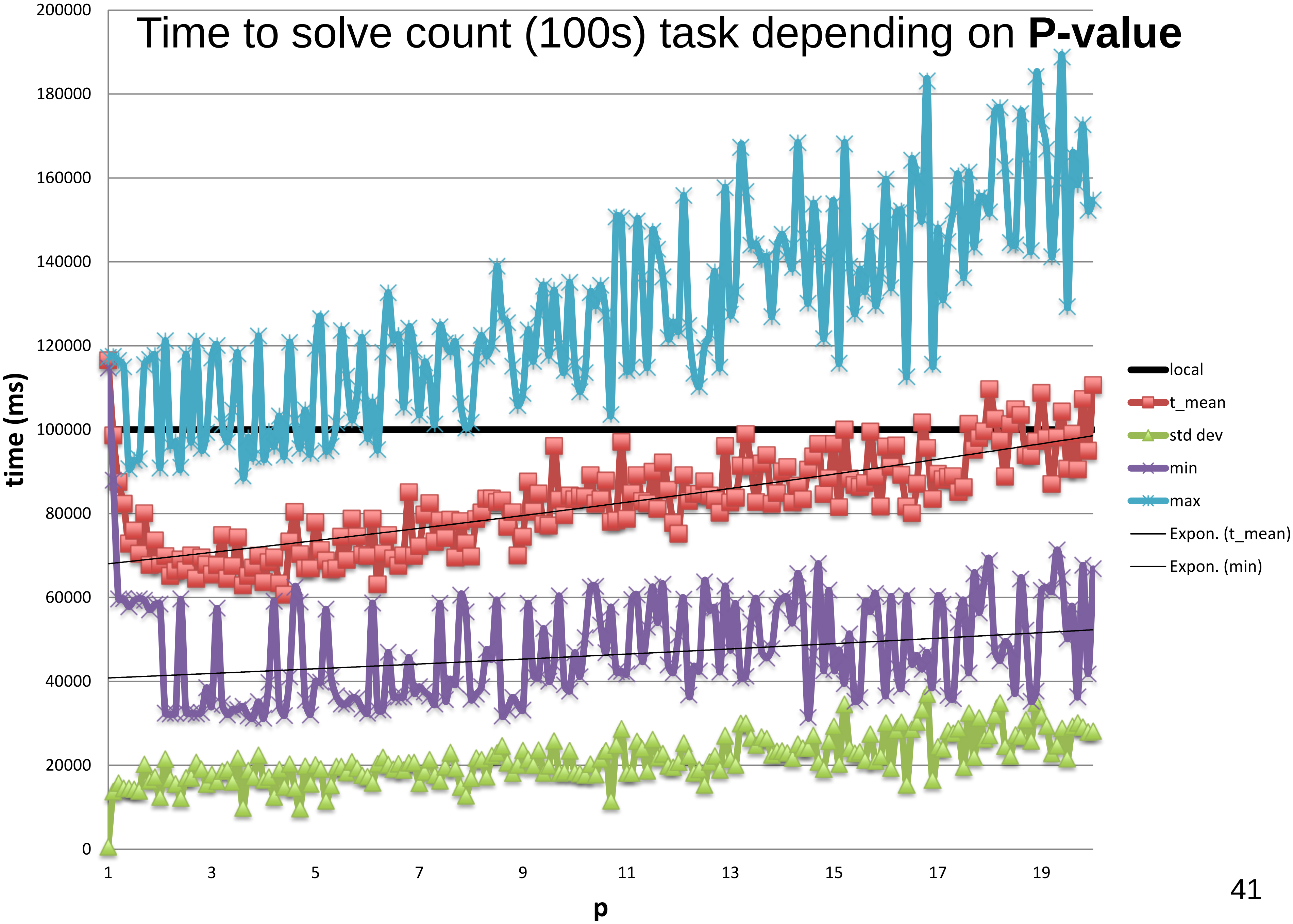


Time to solve count (100s) task depending on **P-value**



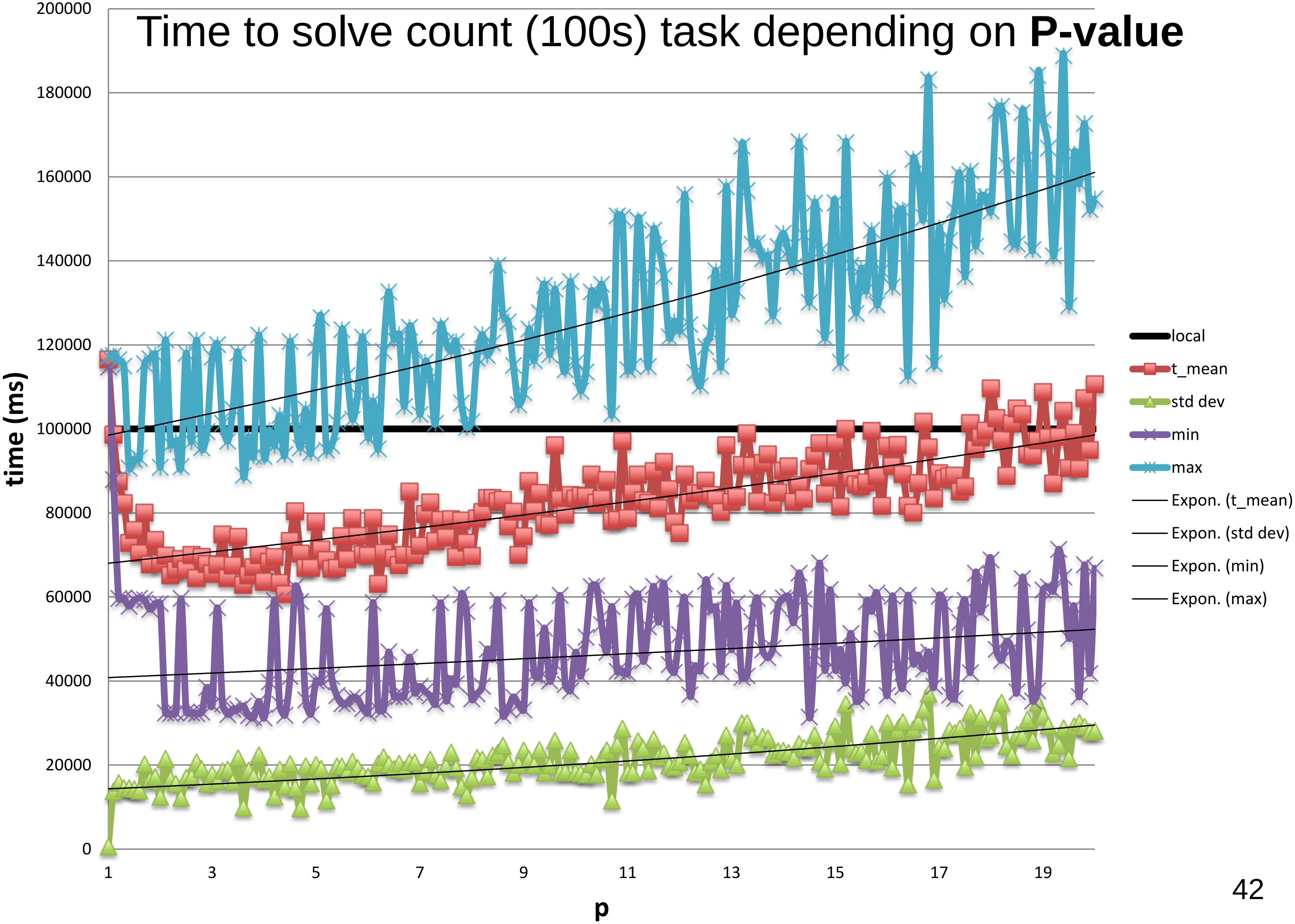
Time to solve count (100s) task depending on **P-value**





Time to solve count (100s) task depending on **P-value**
200000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
time (ms)
1
3
5
7
9
11
13
15
17
19
p
local
t_mean
std dev
min
max
Expon. (t_mean)
Expon. (min)

Time to solve count (100s) task depending on **P-value**



Evaluation

- Phase 1, session 2:

- Conclusions:

  - Mean time, std. dev, min, max all grow with increasing P-value

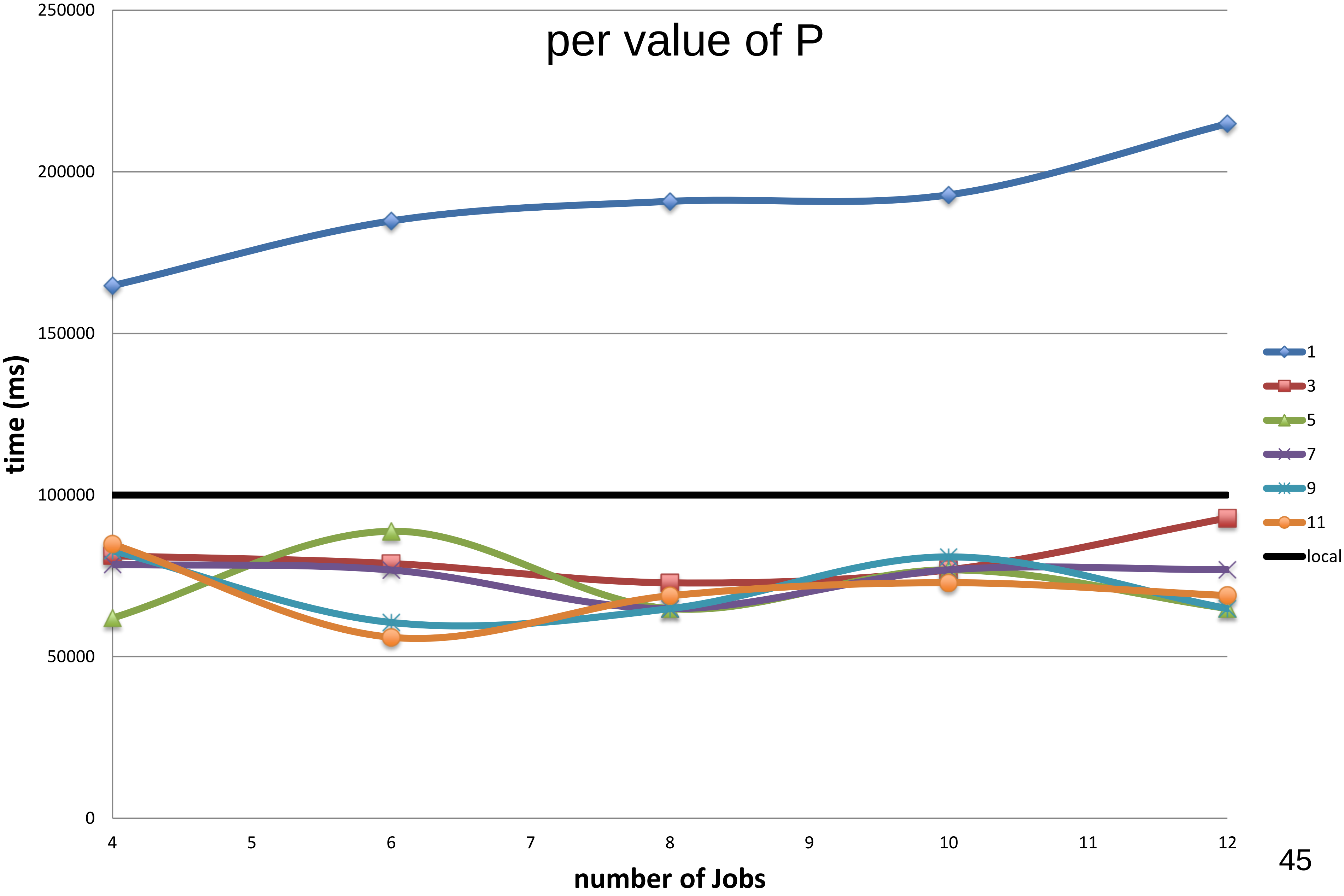
  - Best results are acheived with P-value < 5

  - Might be biased because number of Jobs and number of Providers were constant

Evaluation

- Phase 2, session 1:
- Count task implemented in CCNx/NextServe environment
- Constant inputs for
  - Number of Providers => 4
- Varying inputs for
  - Number of Jobs => {4, 6, 8, 10, 12}
  - Value for P => {1, 2, ..., 11, 12}
- 2 measurements for each set of inputs

Mean time to solve task depending on amount of **Jobs**, per value of P



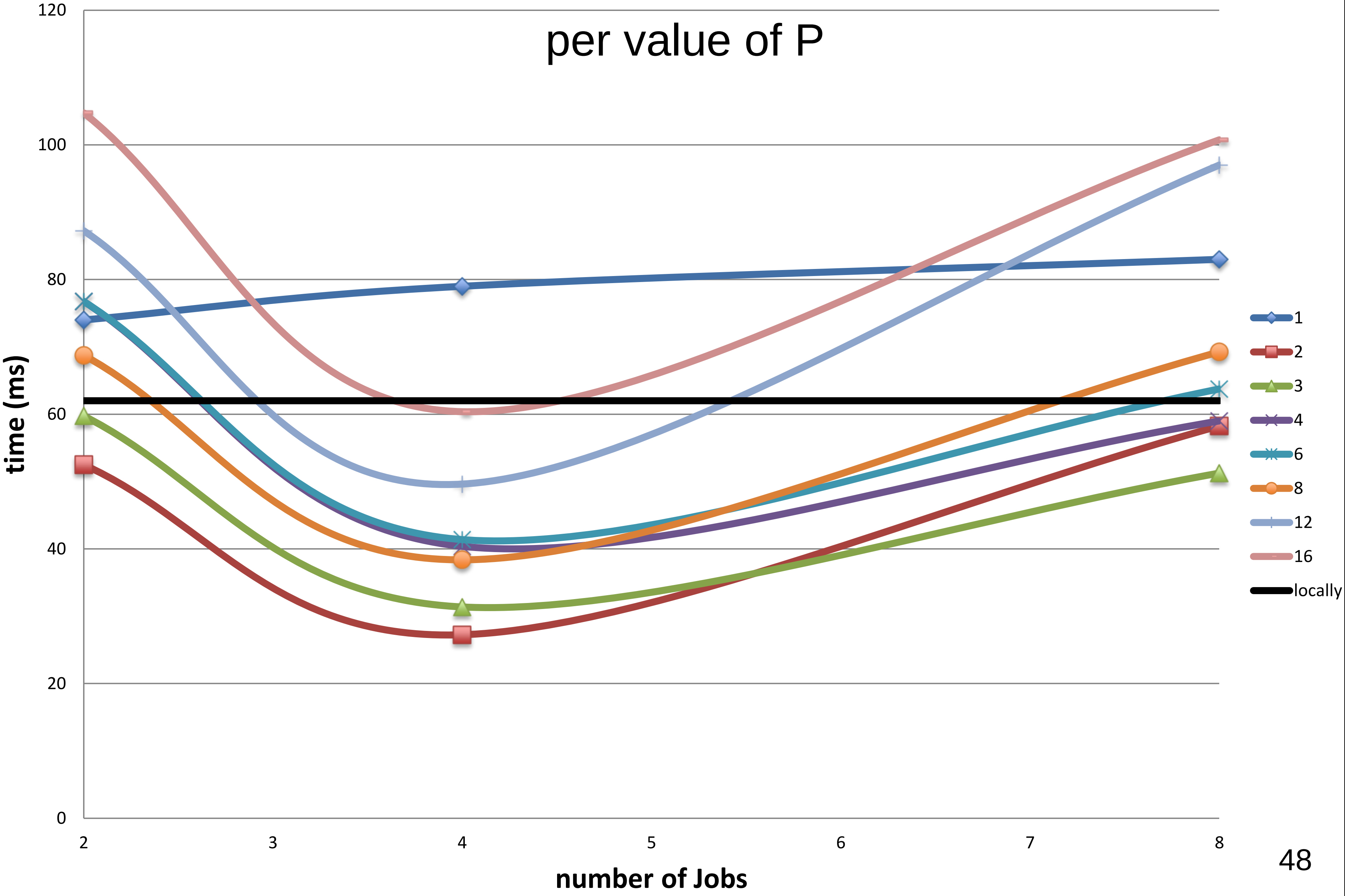
Evaluation

- Phase 2, session 1:
- Conclusions:
  - Qualitatively, measurements in CCNx/NextServe environment seem to match measurements obtained by the PBA-Model
  - More overhead in CCNx/NextServe than in the PBA-Model
  - PBA still proves to be more efficient than to locally process the task

Evaluation

- Phase 2, session 2:
- Transcode a 10 minute video file from 1080p .avi to 576p .mpg in CCNx/NextServe environment
- Task involves transfer of big quantities of data (video files)
- Constant inputs for
  - Number of Providers => 4
- Varying inputs for
  - Number of Jobs => {2, 4, 8}
  - Value for P => {1, 2, 3, 4, 6, 8, 12, 16}
- 2 measurements for each set of inputs

Mean time to solve task depending on amount of **Jobs**, per value of P



Evaluation

- Phase 2, session 2:

- Conclusions:
  - PBA passes the „real-world“ test, still more efficient than locally transcoding
  - Have to be careful with parameters (number of Jobs, P-value)
  - Overhead is rather large (tranfer of video files)

Conclusion

- Possible to distribute services in an ICN network
- Implementation not trivial
  - Need to alter routing/scheduling behaviour of ICN
- It is possible to implement the PBA in CCNx/NextServe
- It is possible to improve turnaround-time for a „real-world“ task (transcoding of a video file) using an implementation of the PBA in CCNx/NextServe

Future work

- CCNx could be extended to support the PBA natively
- Selection of P-Value at run-time could be improved
  - Use information obtained in previous iterations
    - Intervals at which the results arrived
    - Duration after which the results arrived

Thank you!

- Questions?