Simulation of a closed load system

(Assignment 2)

Group members:
Jatin Lachhwani – 21Q050005
Sumon Nath – 21Q050007

System details

Hardware specifications

- Threadpool size: 45
- Number of cores: 4
- Buffer size: 100
- Quantum time: 10 ms
- Context switch time: 0.2 ms

Classes

- Request
- RequestBuffer
- Threadpool
- Core
- Event
- EventQueue
- System
- Simulation

Classes

Class: Request

Attributes

- requestId: int
- threadId: int
- arrivalTime: double
- remainingTime: double
- serviceTime: double
- ThinkTime: double
- TimeoutTime: double

Class: RequestBuffer

Attributes

- Size: double
- Buffer: queue<reqId>

Methods

- Enqueue
- Dequeue

Class: ThreadPool

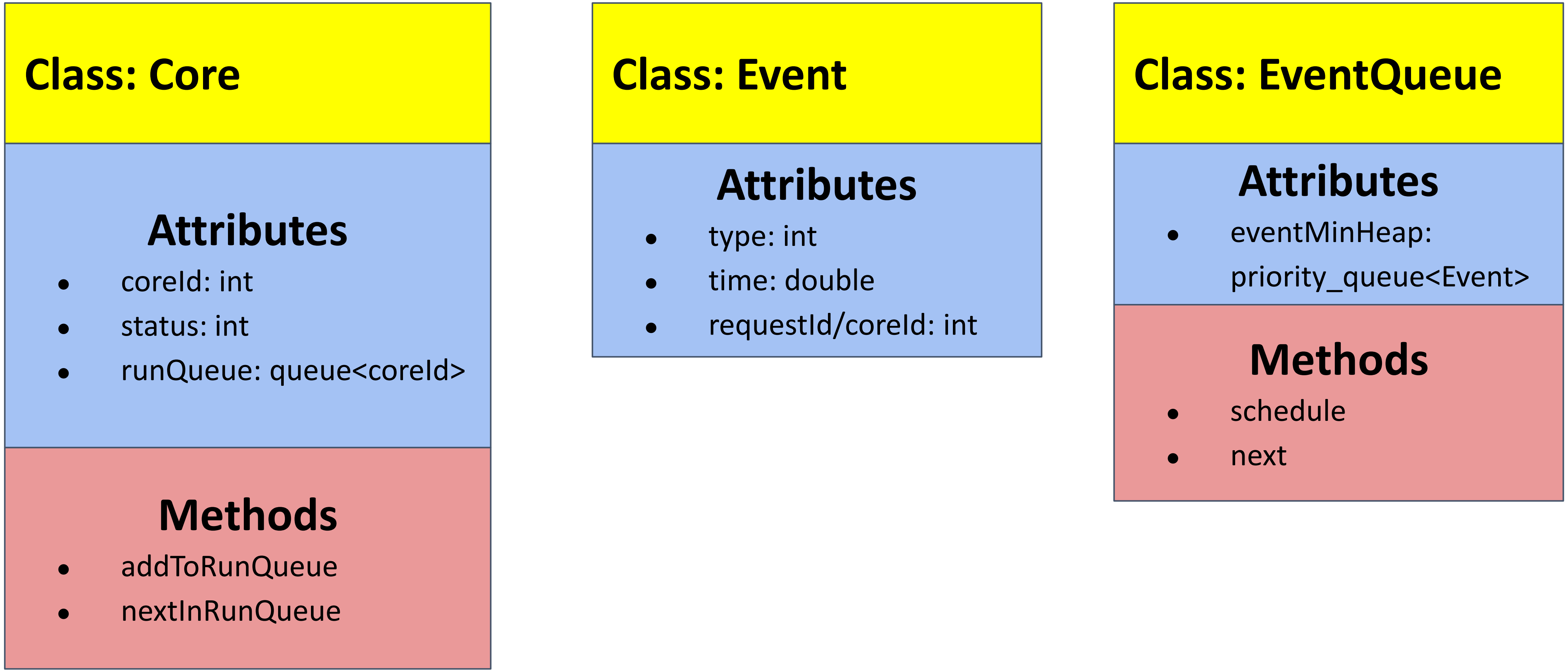
Attributes

- threadPool: vector<threadId>

Methods

- assignThread
- releaseThread

Classes



Class: Core

Attributes

- coreId: int
- status: int
- runQueue: queue<coreId>

Methods

- addToRunQueue
- nextInRunQueue

Class: Event

Attributes

- type: int
- time: double
- requestId/coreId: int

Class: EventQueue

Attributes

- eventMinHeap: priority_queue<Event>

Methods

- schedule
- next

Classes

Class: System

Attributes

- threadPool: ThreadPool
- requestBuffer: RequestBuffer
- cpu: vector<Core>
- numberOfCores: int

Class: Simulation

Attributes

- simTime: double
- lastEventTime: double
- eventQueue: EventQueue
- requestList: vector<Request>
- numOfCompletion: int
- successes: int
- timeOuts: int
- drops: int
- accumulatedResponseTime: double
- areaServerStatus: vector<double>

Methods

- resetStats
- initialize
- timing
- updateStats
- printMetrics
- onArrival
- onContextSwitch
- onPreemption
- onDeparture

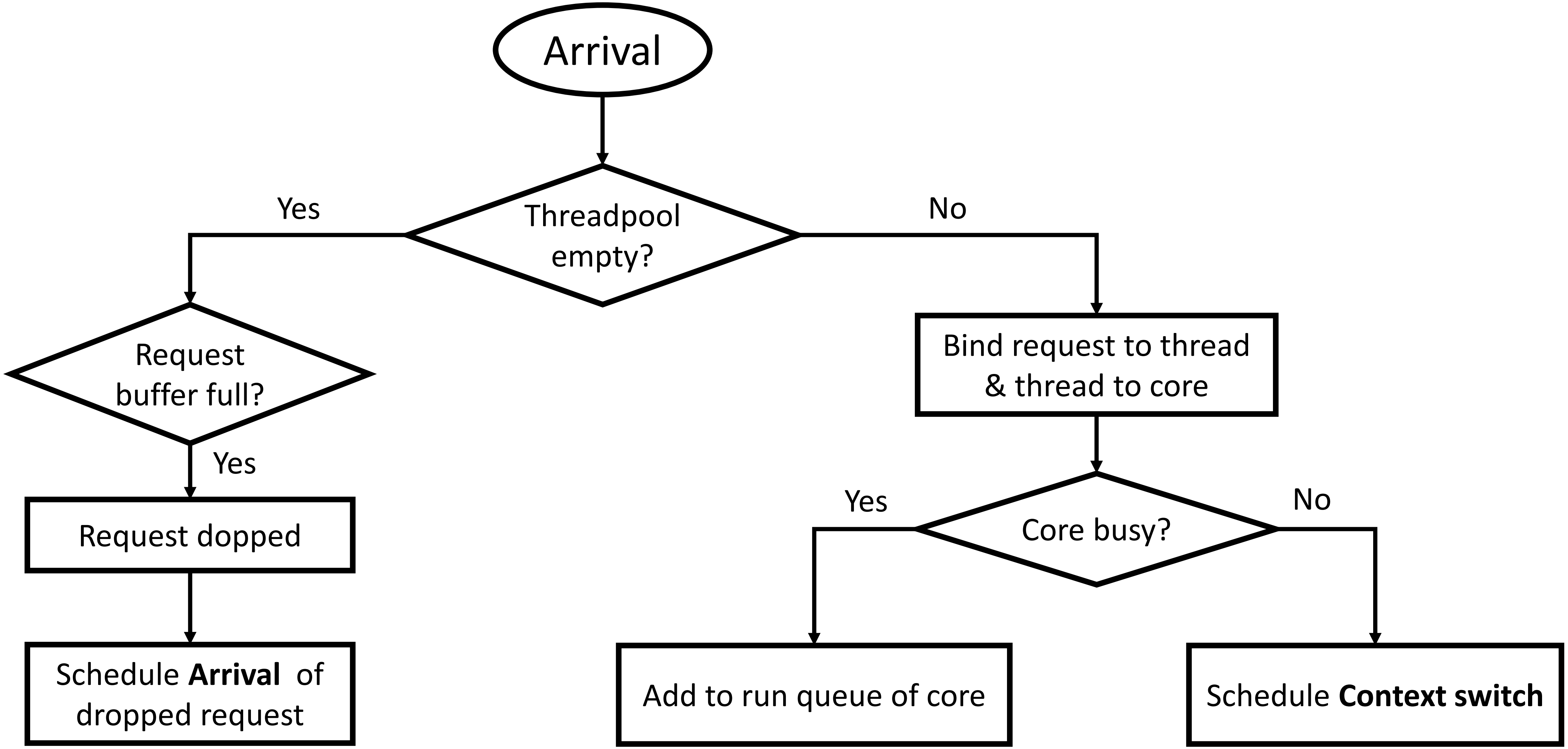
Type of Events

- Arrival
- Context Switch
- Preemption
- Departure

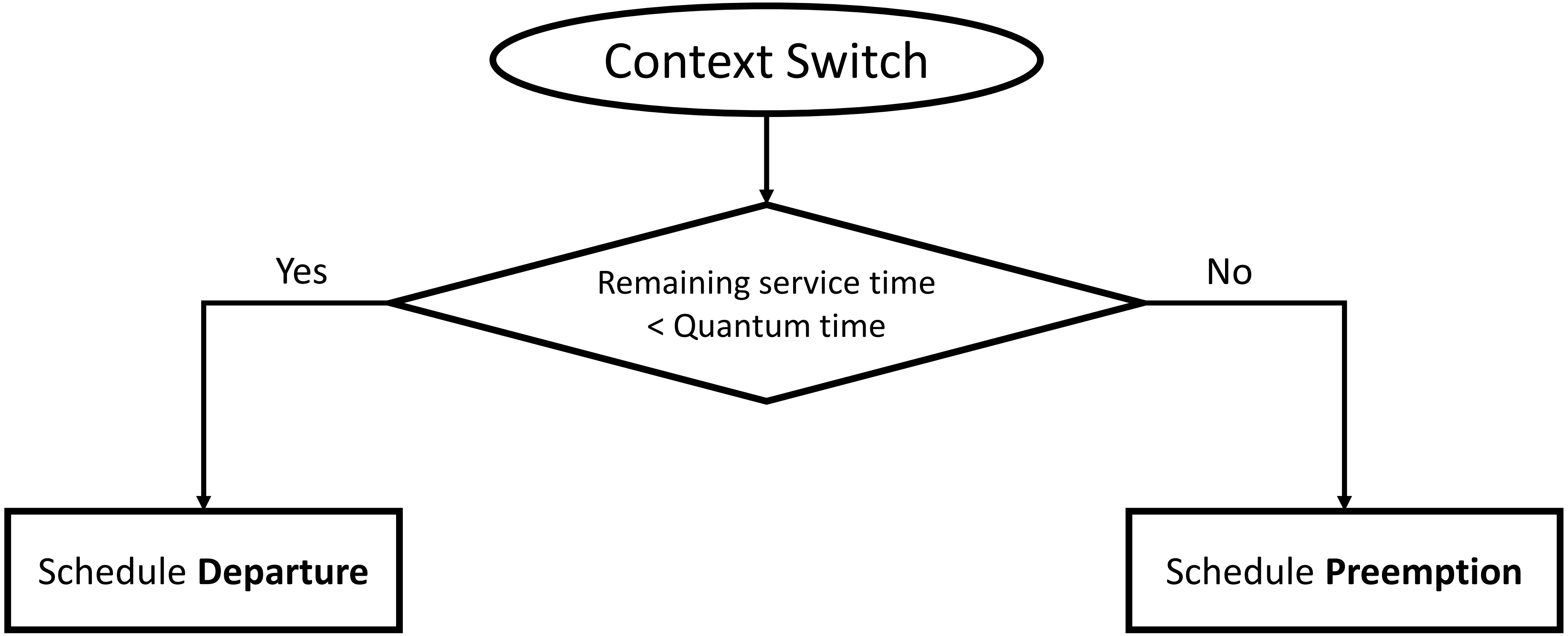
Event Handlers

- onArrival
- onContextSwitch
- onPreemption
- onDeparture

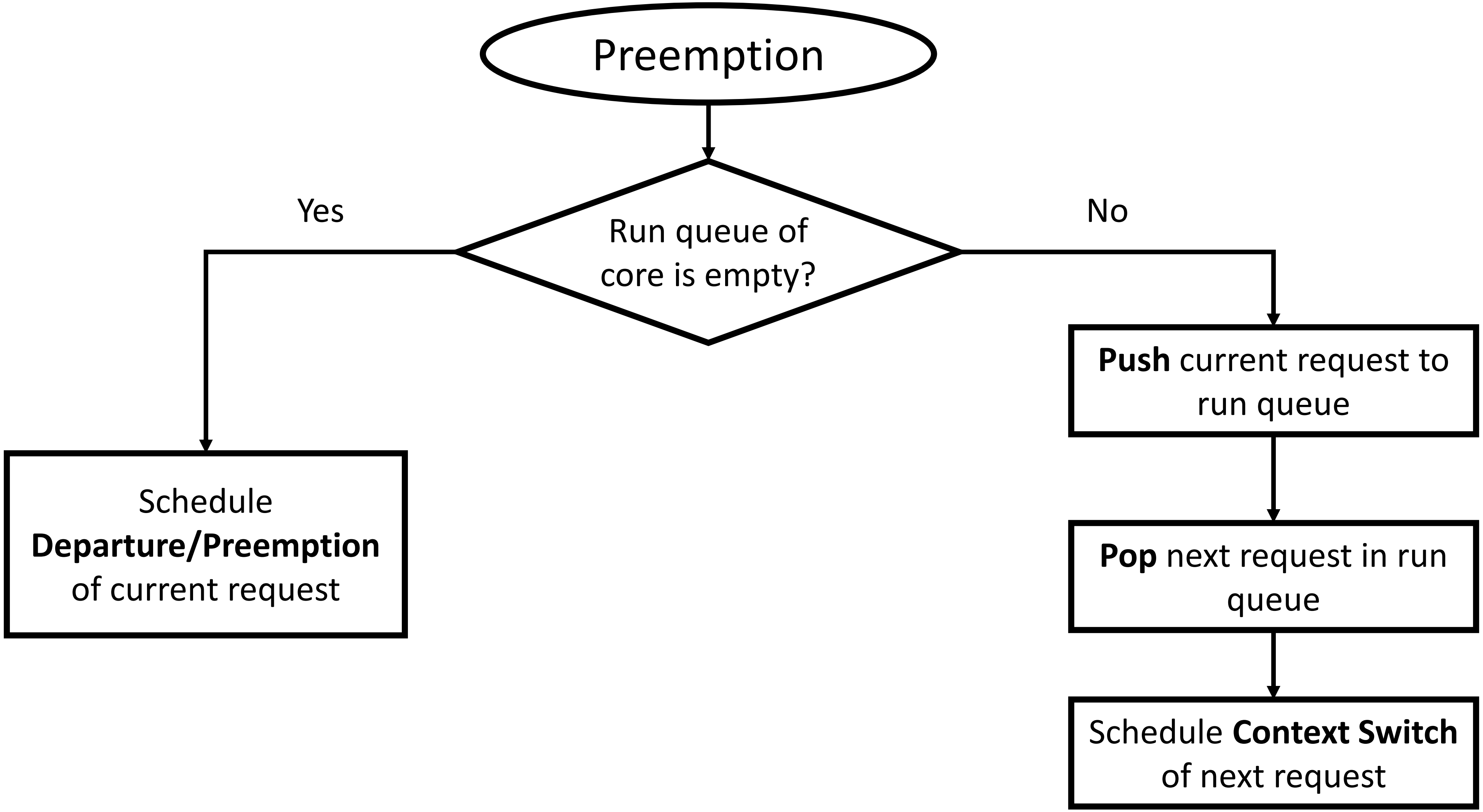
On Arrival


Arrival
Threadpool empty?
Yes
No
Request buffer full?
Yes
Request dopped
Schedule **Arrival** of dropped request
Bind request to thread & thread to core
Core busy?
Yes
No
Add to run queue of core
Schedule **Context switch**

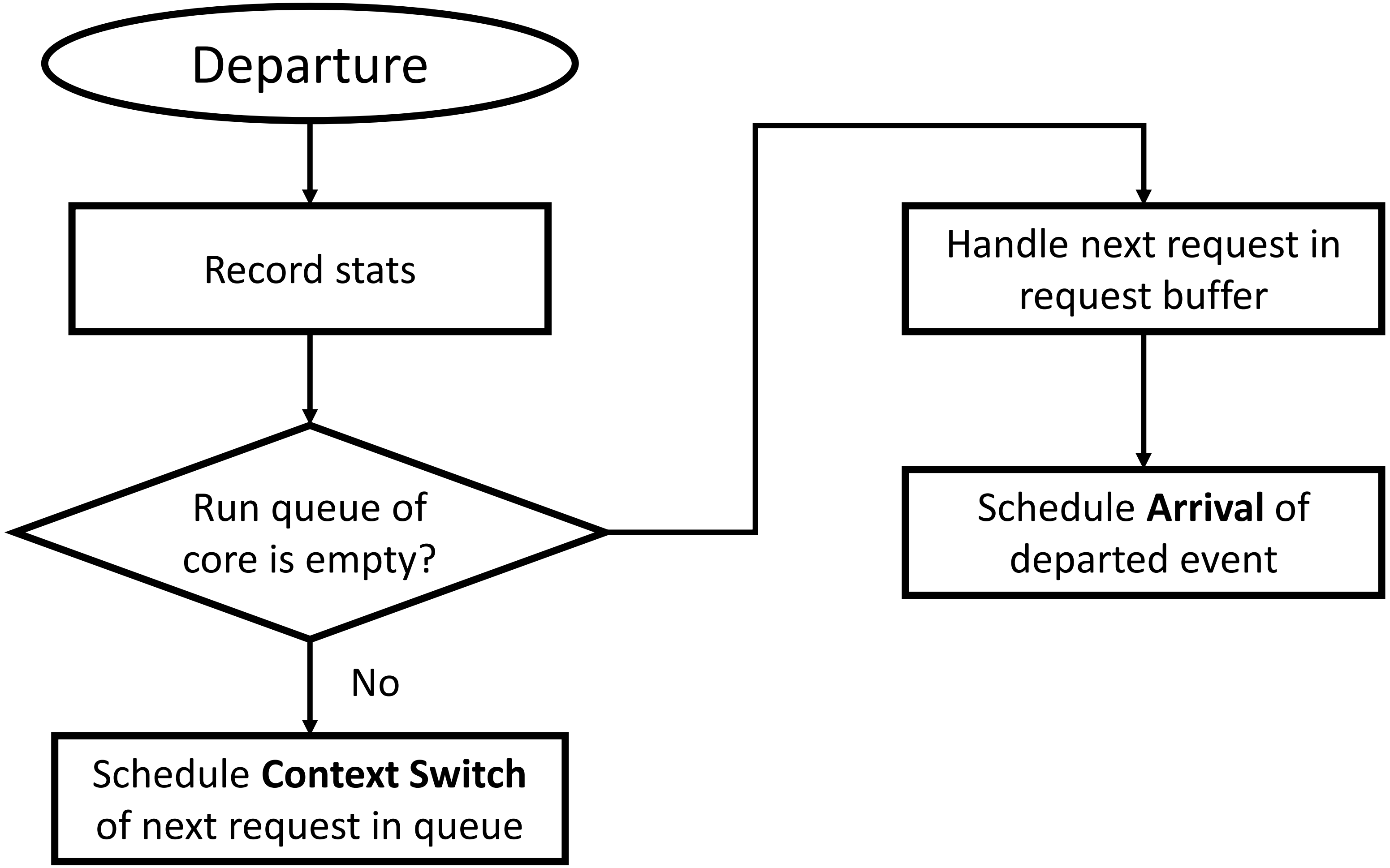
On Context Switch



On Preemption


Preemption
Run queue of core is empty?
Yes
No
Schedule **Departure/Preemption** of current request
Push current request to run queue
Pop next request in run queue
Schedule **Context Switch** of next request

On Departure



Departure

Record stats

Run queue of core is empty?

No

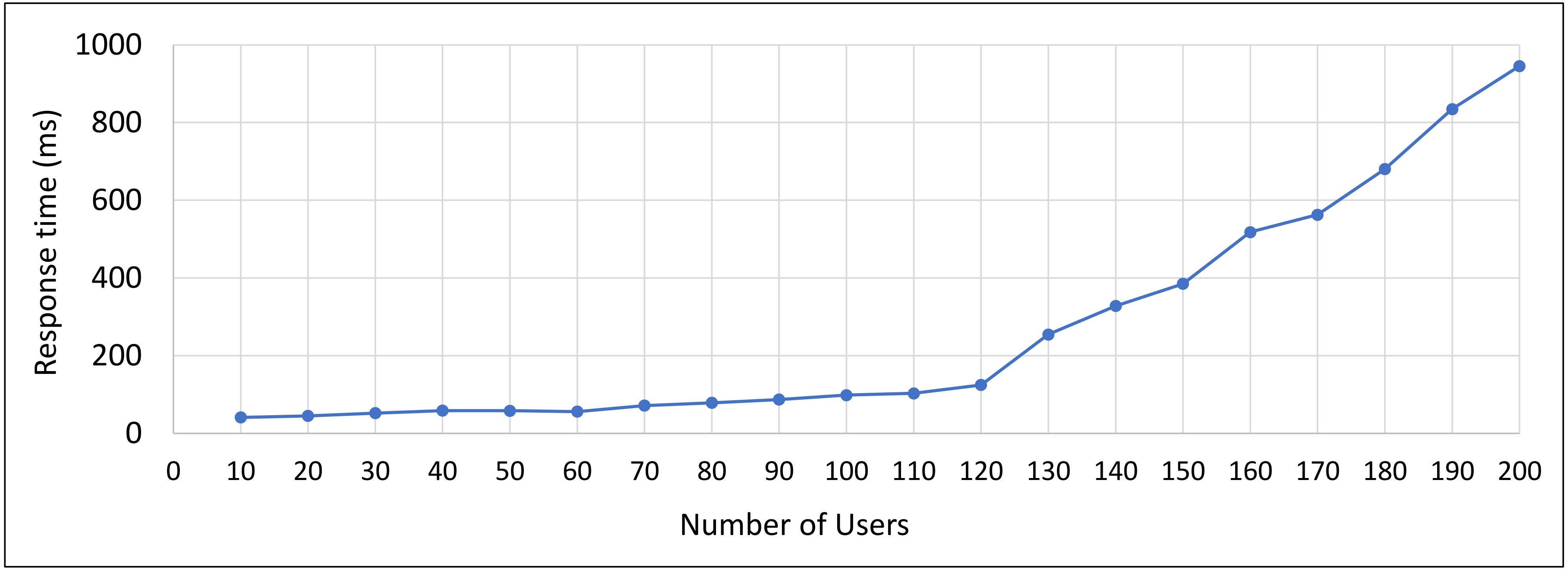
Schedule **Context Switch** of next request in queue

Handle next request in request buffer

Schedule **Arrival** of departed event

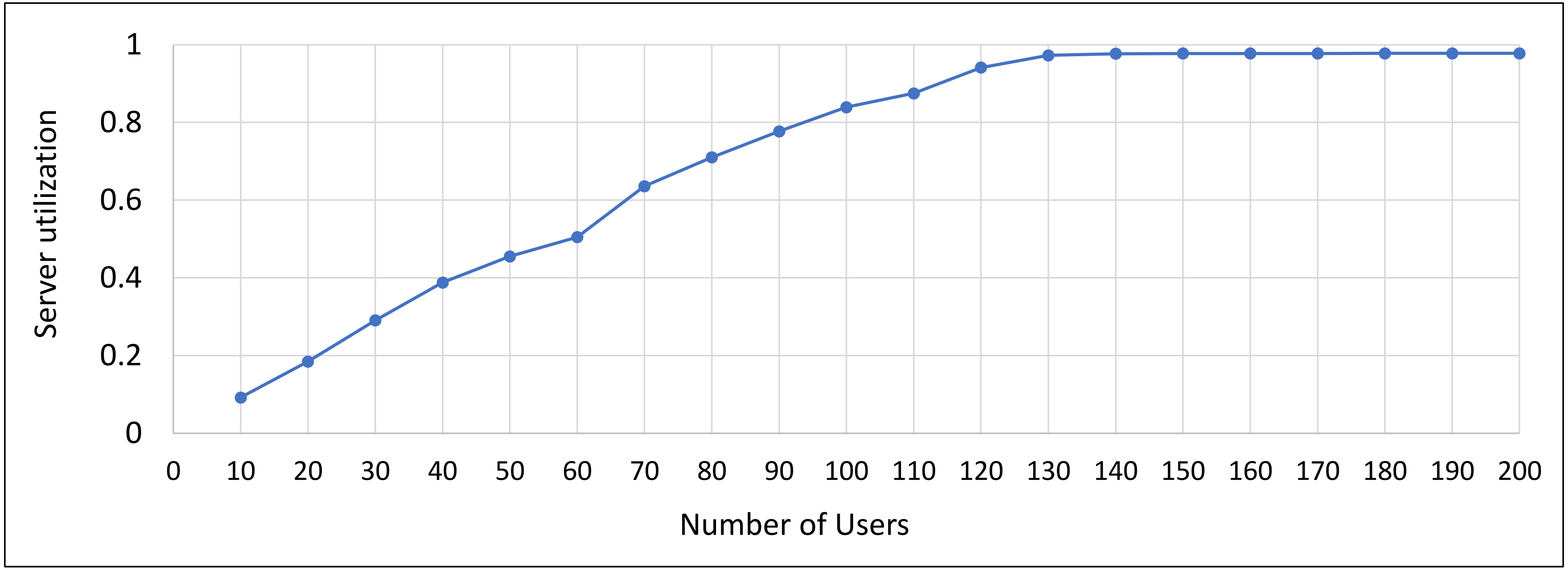
Simulation Analysis

Response time vs number of users



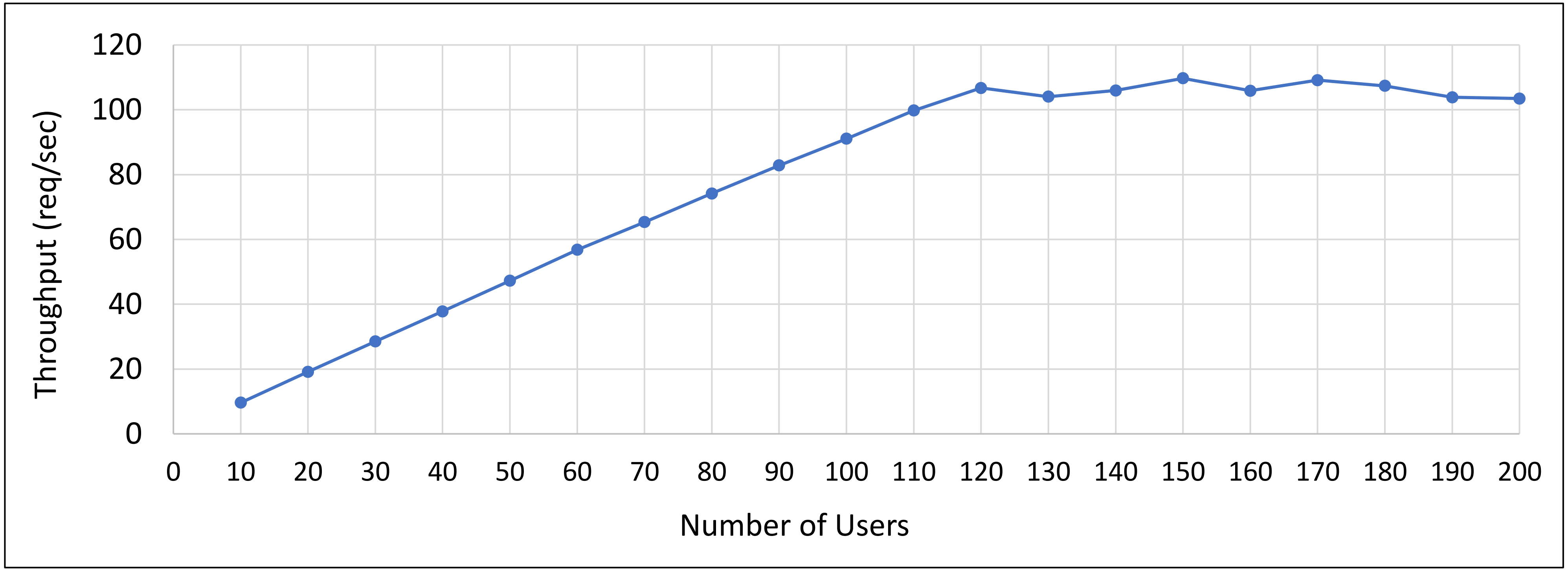
- The mean service time of each request is 50 ms, and are extracted from an exponential distribution
- The average response time before 100 users is ~68 ms. Due to the overheads like context switching, busy cores the response time is more than the service time
- Beyond 100 users the response time overshoots where the server reaches its limit.

Server utilization vs number of users



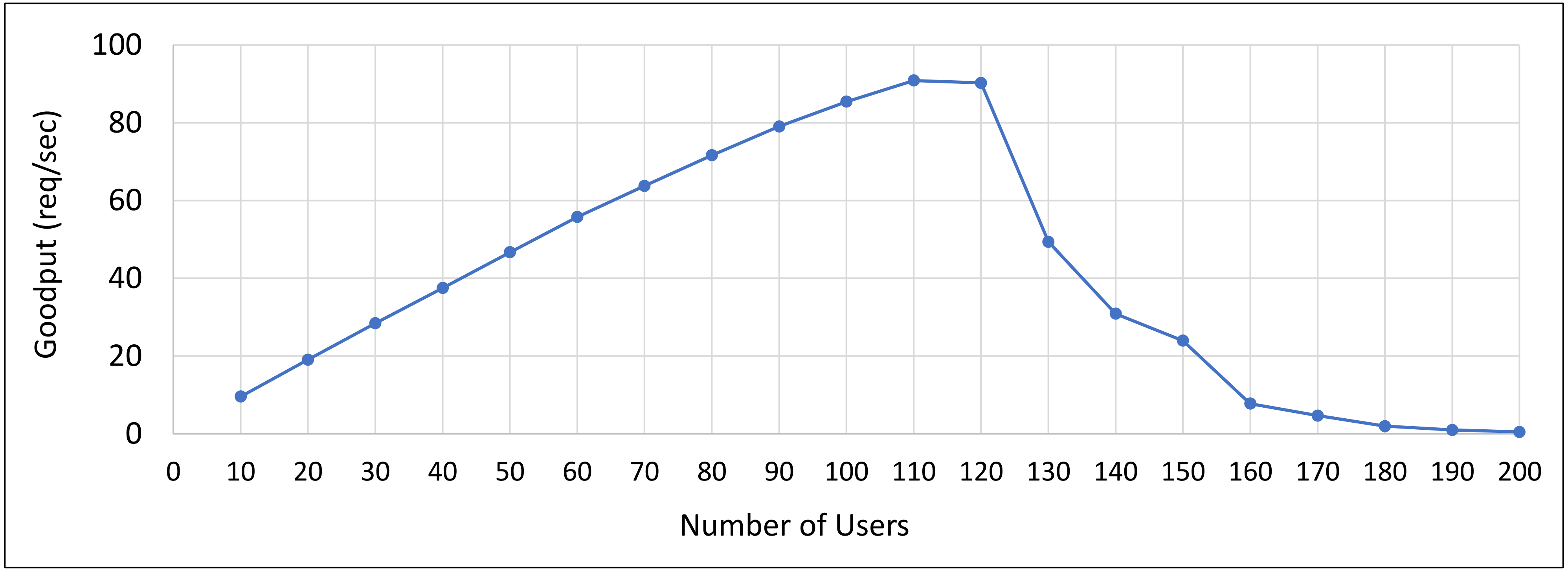
- At about 120 users the server is saturated

Throughput vs number of users



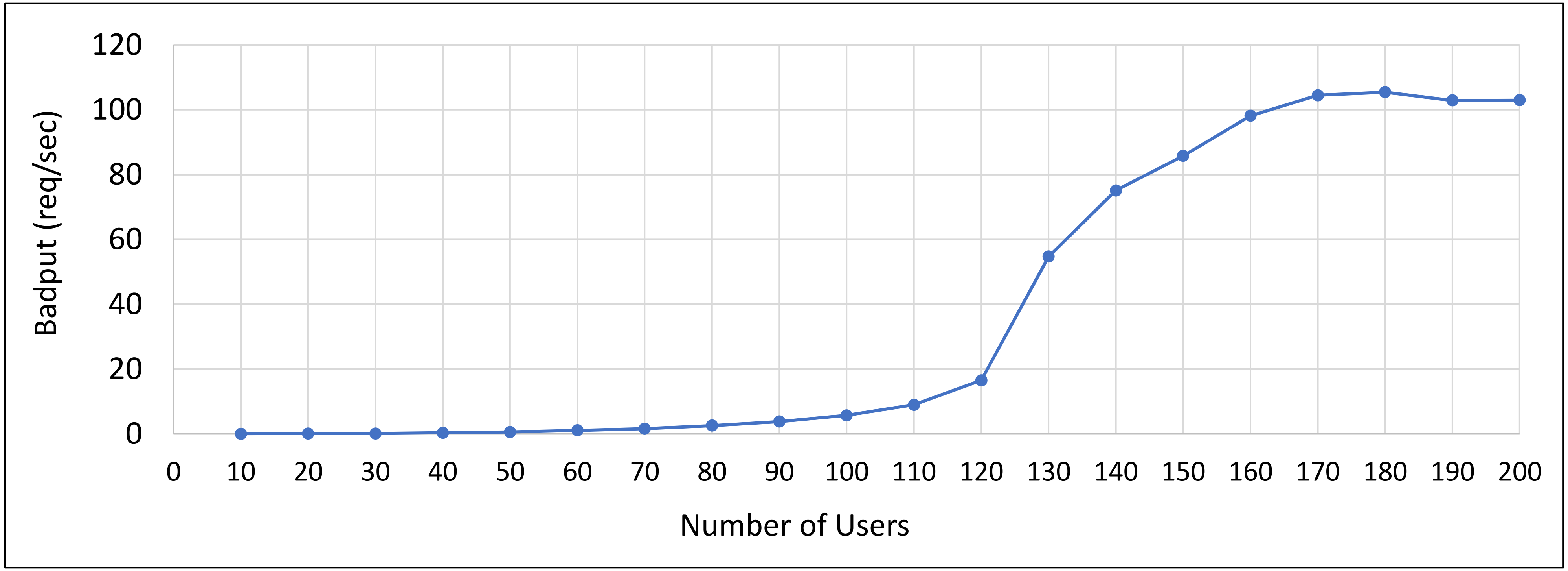
- Before reaching the maximum numbers of users supported, the throughput increases linearly with the number of users as expected.
- After 120 users, the server saturates and the throughput remains constant at ~105 req/sec.

Goodput vs number of users



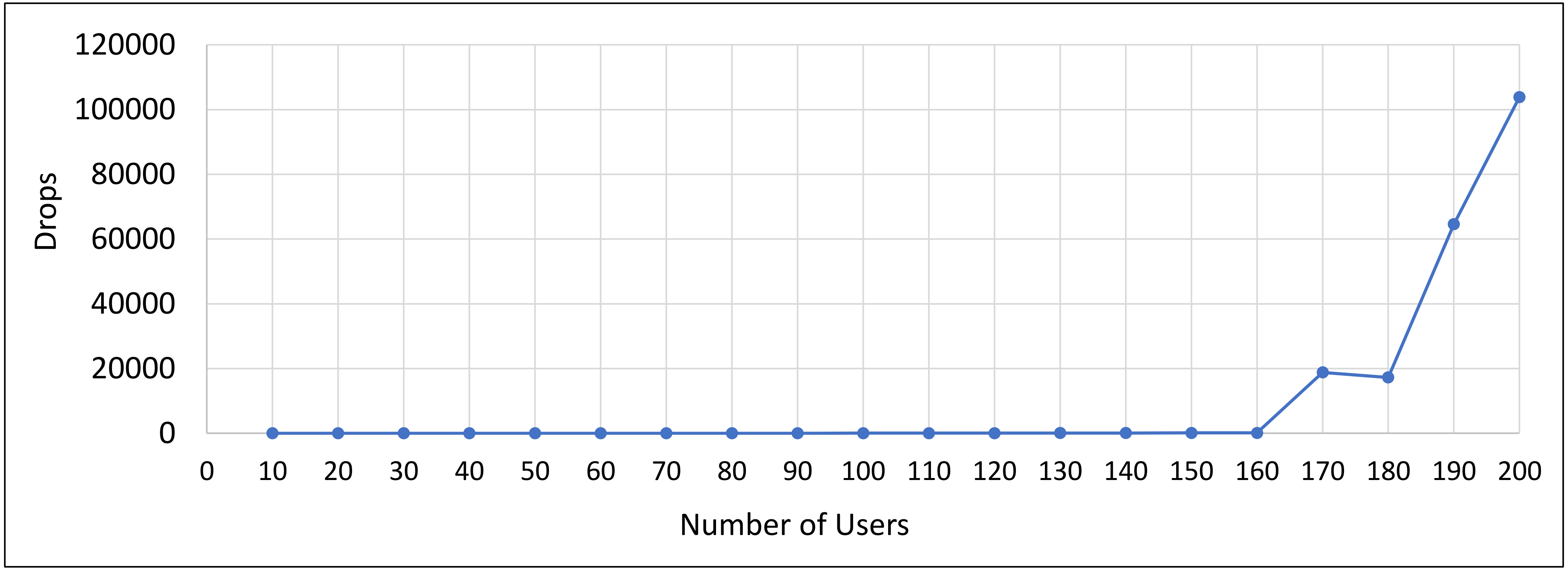
- Before the system reaches saturation number of users = 120, Goodput = Throughput as requests are timeout.
- After the saturation point the Goodput drops drastically which signifies that a large fraction of the requests are timed out.

Badput vs number users



- Before the system reaches saturation number of users = 120, Badput is almost 0 req/sec as no requests are timeout.
- After the saturation point the Badput increases sharply as more and more request are timed out due to increased congestion as the server.

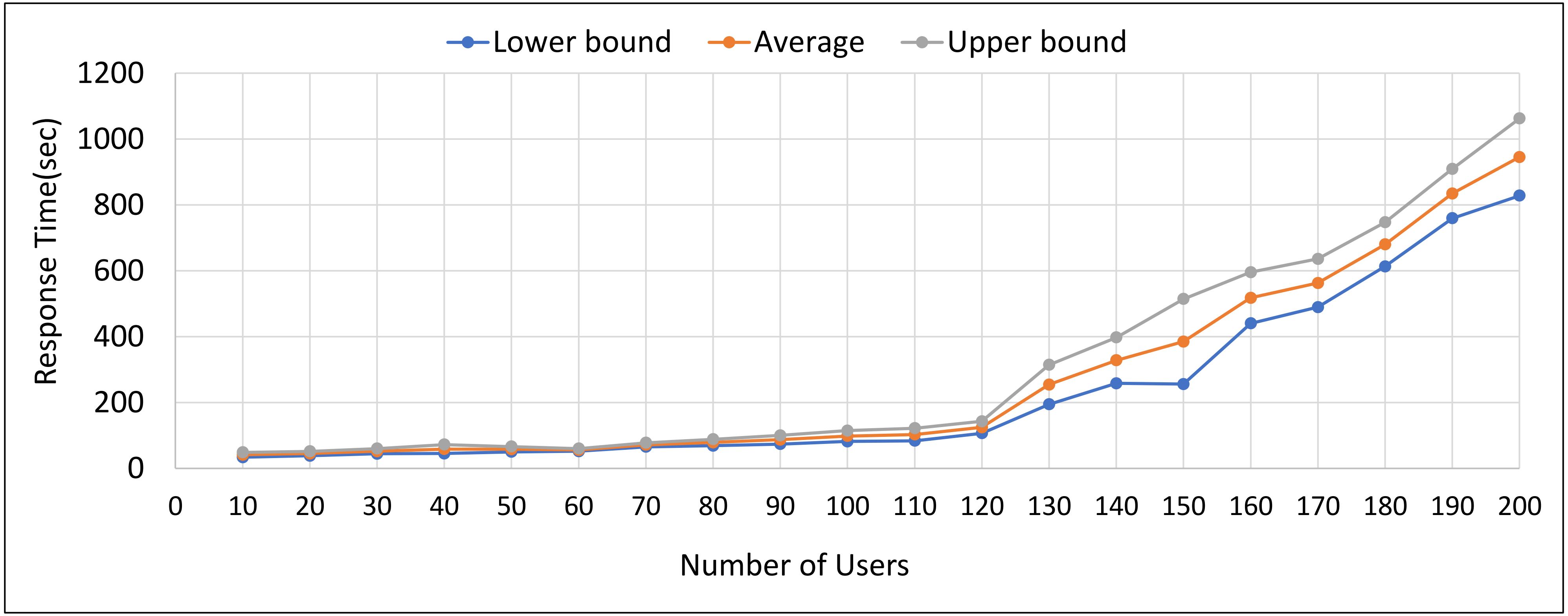
Drops vs number users



- After the saturation point the number of drops increases drastically due to increased contention in the request buffer.

Confidence interval

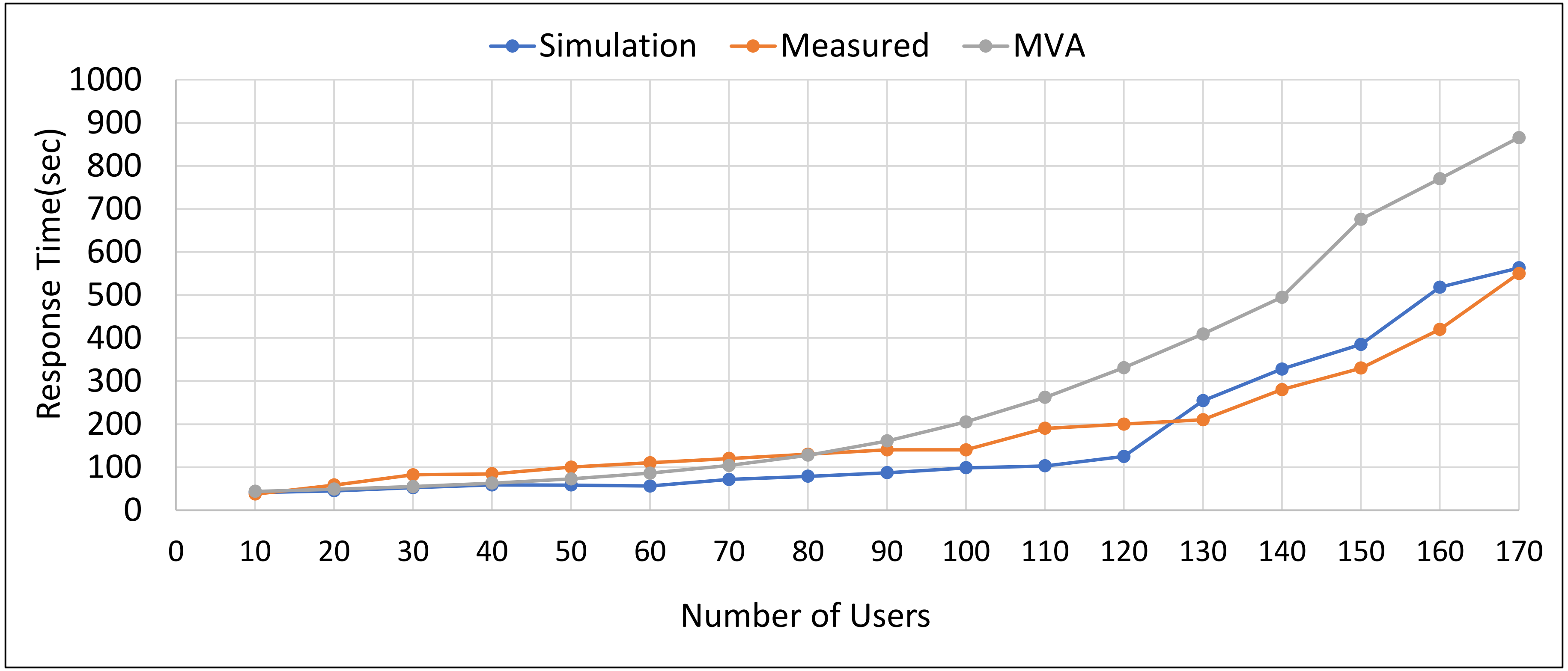
Response time vs number users



- With 95% confidence we can say that the Response time for corresponding Users is bound in the following interval.

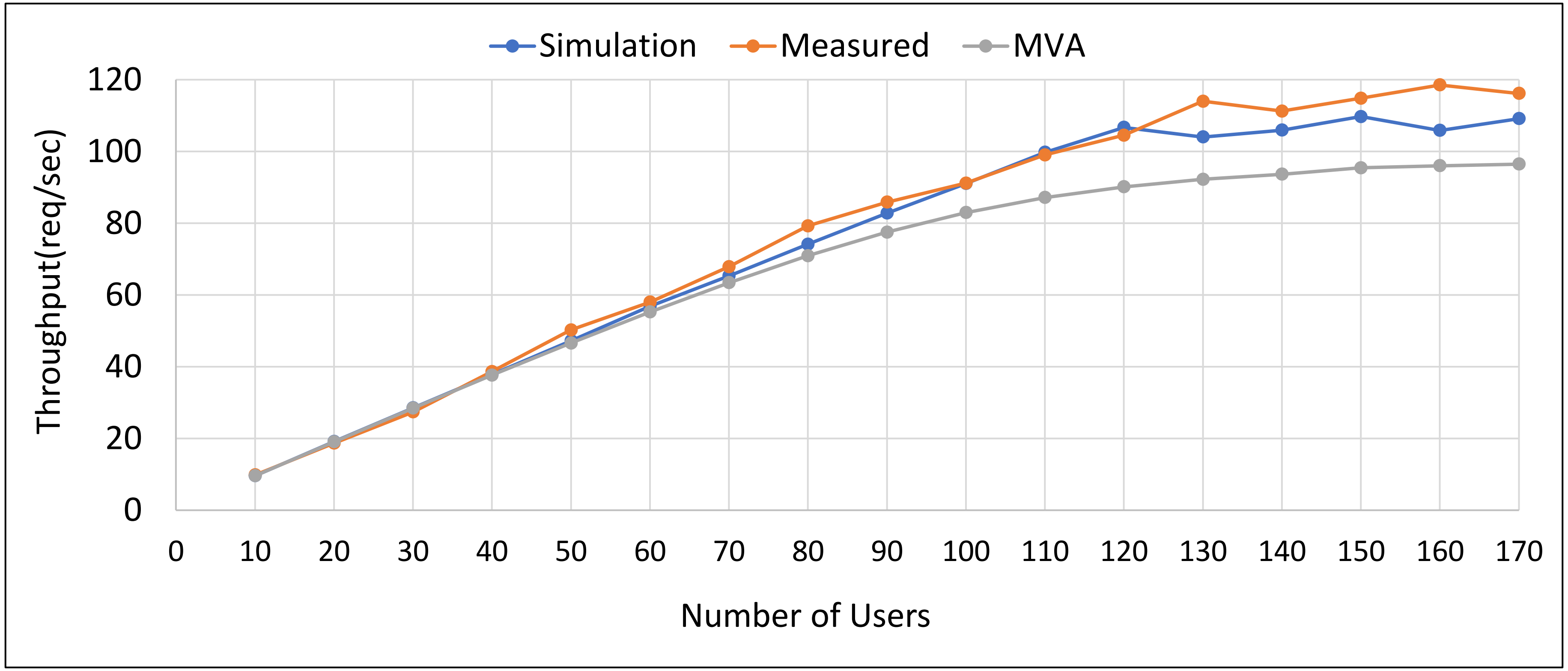
Comparison: MVA, measured, simulation

Response time vs number users



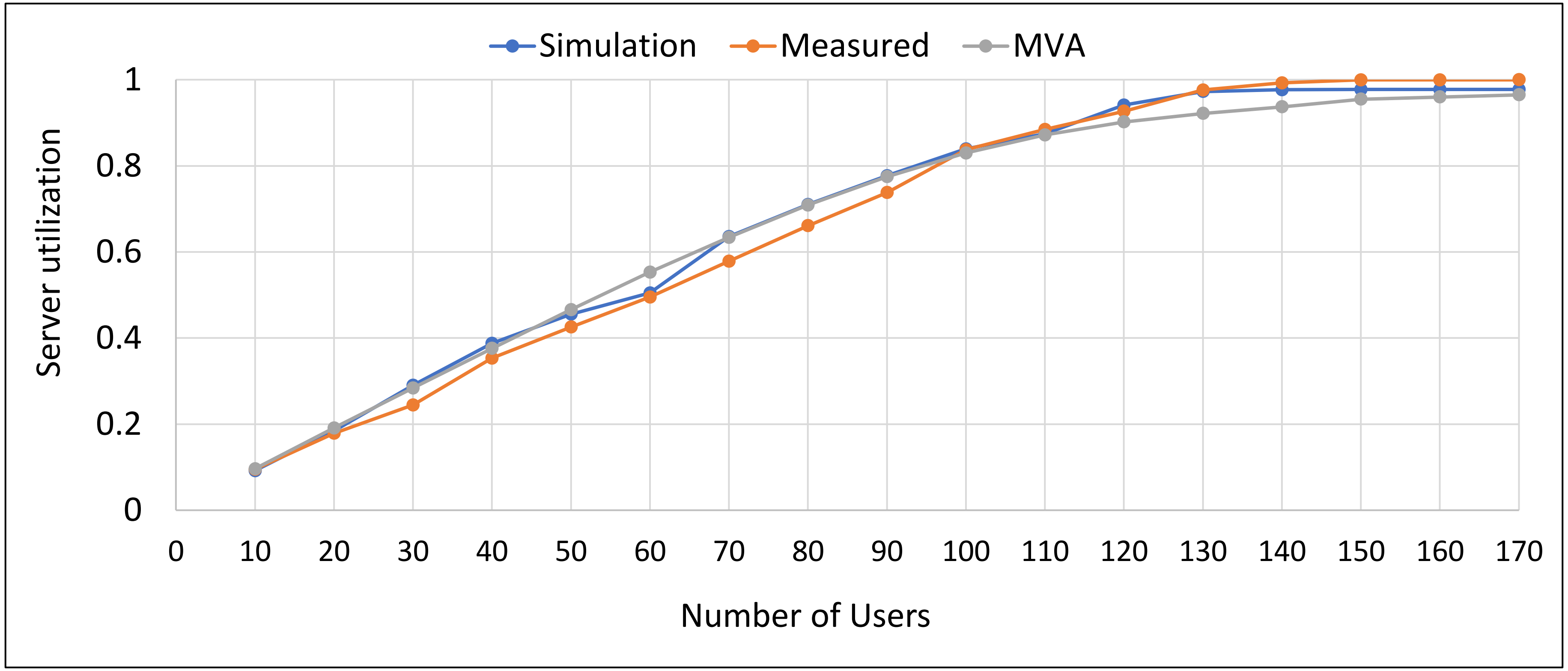
- As expected the response time is lower incase of simulation as we do not consider many factors that can lead to delays in a real system.

Throughput vs number users



- The throughput is almost same in both simulation and measured analysis
- For MVA we get a smoother graph as it does not take randomness into account

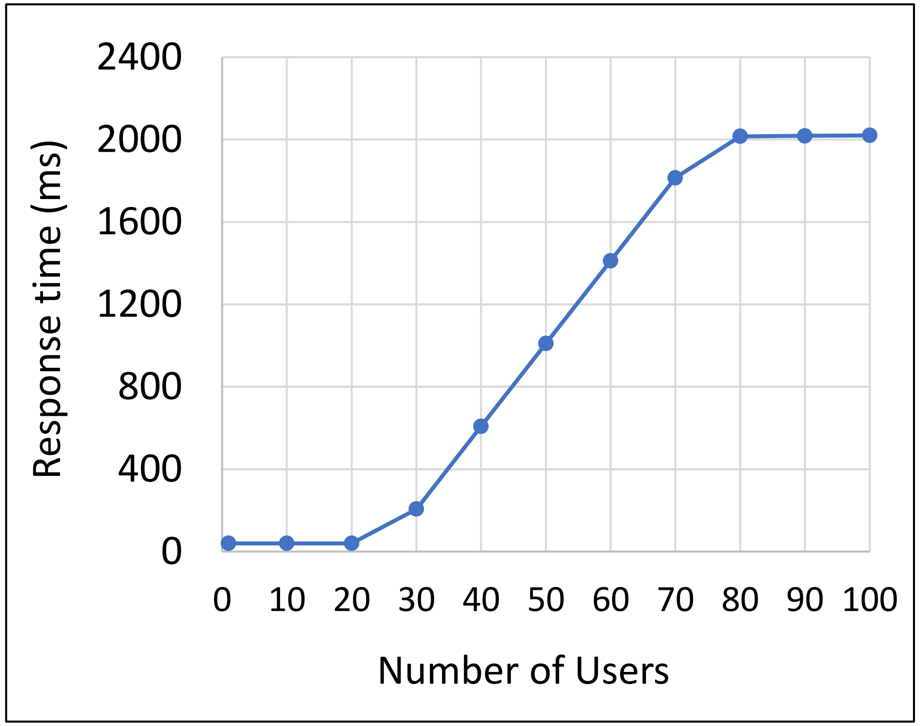
Server utilization vs number users

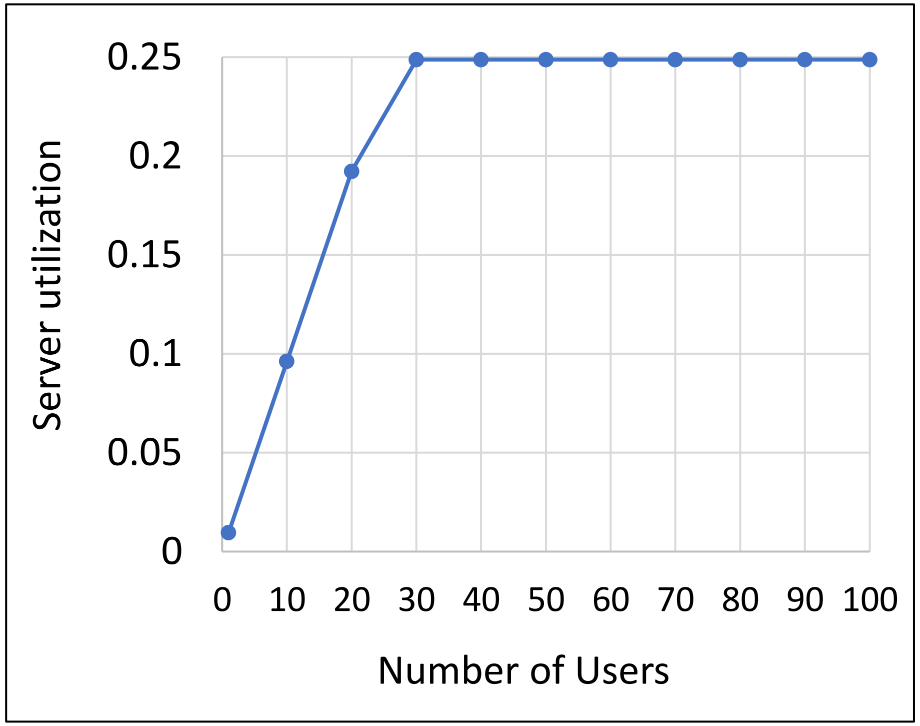


- The server utilization is also close which is obvious as the throughputs are almost same

More insights

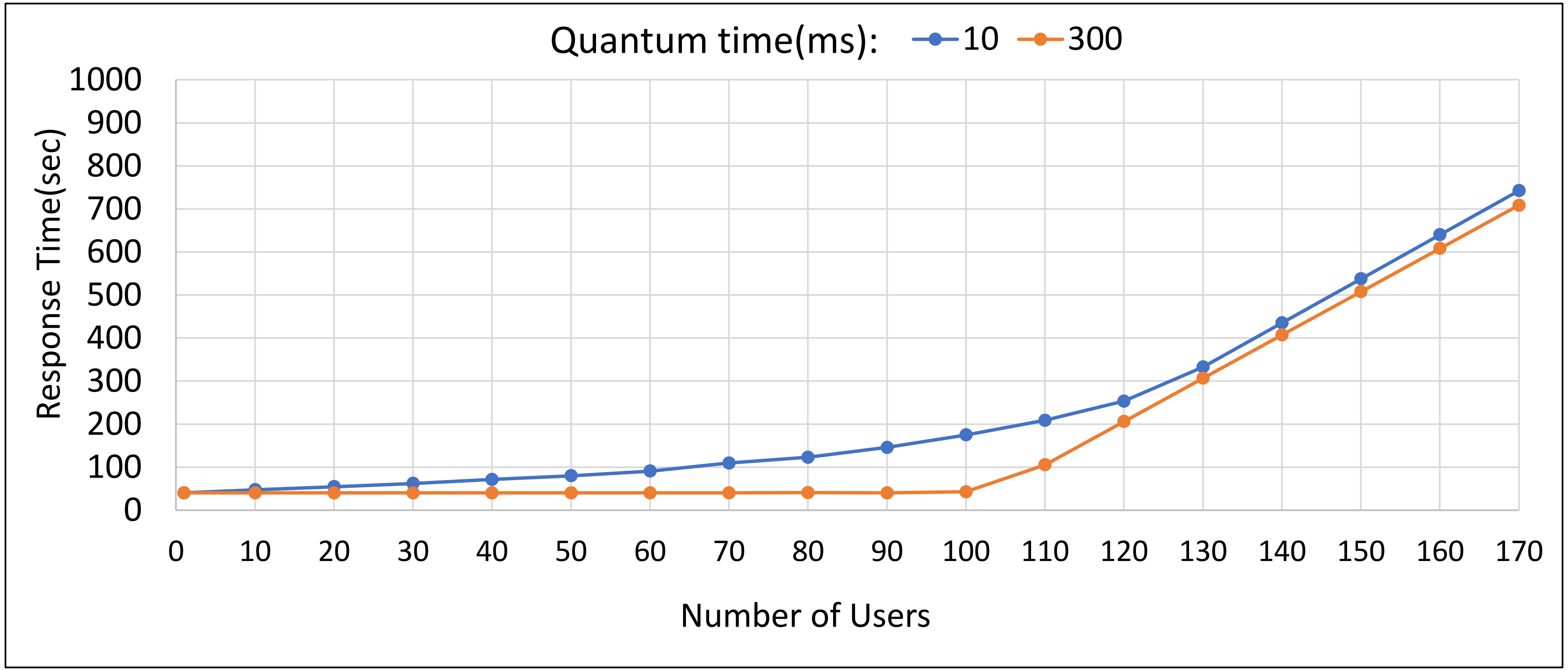
Number of threads = 1





- A single thread only allows one core to be active. Hence the utilization peaks at 0.25.
- This simulates a system with a single core, even though 4 are available only 1 is used.

Varying quantum time



- As the quantum time is increased to 300 ms, the response time remains constant at about 40.2 ms which is same as the service time of 40 ms + the initial context switch time of 0.2 ms.

Thank You