IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 15, Issue 2 Ser. I (Mar – Apr 2019), PP 85-91
www.iosrjournals.org

Comparative Study of Single vs Batch Service Policies in M/M/C Queueing Systems with Breakdown and Repair Mechanisms

K.P.S. Baghel
Government Degree College, Targawan Jaithra, Etah (UP)

Abstract
Choosing between single and batch service policies in multi-server queueing systems is a decision with far-reaching consequences for system throughput, customer waiting times, and operational cost. This article presents a comparative analysis of single and batch service disciplines within the M/M/C queueing framework, extended to incorporate server breakdowns and repair mechanisms — two features that are conspicuously absent from most textbook treatments but absolutely central to real-world system behavior. We examine how each service policy responds to server unreliability, analyzing steady-state performance metrics including mean queue length, system throughput, server utilization, and expected waiting time under both disciplines. The role of breakdown rates, repair rates, and batch size distributions in shaping these metrics receives detailed attention. We discuss the conditions under which batch service outperforms single service and vice versa, and identify the parameter regimes where the difference between policies is most consequential. Approximation methods, exact analytical results, and simulation-based validation are all addressed. The article concludes with practical guidance for system designers choosing between these service disciplines under realistic operational constraints.

Keywords: *M/M/C queue, batch service, repair mechanism, performance comparison, server breakdown, single service*

I. Introduction

Anyone who has stood in a supermarket checkout line has an intuitive feel for single service queueing. One customer steps up, the cashier processes their items, they leave, and the next customer steps forward. Clean, sequential, familiar. Now imagine a shuttle bus stop where the bus only departs once it has collected at least eight passengers. You might arrive first and wait considerably longer than the person who arrives just before the bus fills up — even though you have been waiting far longer. That is batch service, and the difference in experience between these two scenarios captures something genuinely important about how service discipline shapes the relationship between waiting and throughput.

In formal queueing theory, M/M/C systems with multiple servers have been studied for decades. The Poisson arrival process (the first M), exponential service times (the second M), and C parallel servers provide a tractable and reasonably realistic model for a wide range of systems — call centers, bank branches, manufacturing workstations, hospital outpatient departments. Within this framework, the choice of service discipline — single versus batch — is a design decision with measurable performance consequences.

What makes this comparison genuinely interesting is the addition of server breakdowns and repairs. A server that occasionally fails is not merely an inconvenience — it changes the fundamental dynamics of the queue in ways that interact differently with single and batch service. When a single-service server breaks down, one customer's service is interrupted. When a batch-service server breaks down mid-batch, an entire group of customers faces disruption. The statistical properties of this disruption, how frequently it occurs, how long repairs take, and whether service resumes where it left off or restarts from scratch, all affect the relative performance of the two policies in ways that are not obvious without careful analysis.

This article works through the comparison systematically. We start with the basic M/M/C model and its extension to batch service, then introduce breakdown and repair mechanics, develop the performance metrics of interest, and finally present a structured comparison of how the two policies behave across a range of operating conditions. The goal is not to declare a winner — the answer genuinely depends on the application — but to give system designers and operations researchers a clear picture of the trade-offs involved.

II. The M/M/C Framework: Single Service Baseline

2.1 Model Structure and Assumptions

The M/M/C queue is a workhorse model for good reason. Customers arrive according to a Poisson process with rate λ, meaning inter-arrival times are exponentially distributed with mean 1/λ. There are C servers,

each with exponentially distributed service times at rate μ. Customers join a single queue and are served by the next available server. The system is stable when the traffic intensity $\rho = \lambda/(C\mu)$ is strictly less than one — otherwise the queue grows without bound.

Under these assumptions, the steady-state distribution of the number of customers in the system has a known closed form, derived from the global balance equations of the underlying Markov chain. The celebrated Erlang C formula gives the probability that an arriving customer must wait — the cornerstone of capacity planning in telecommunications and service operations. From this, mean waiting times, queue lengths, and server utilization all follow in straightforward fashion.

The elegance of this model is precisely why its limitations matter. Real servers break down. Exponential service times may be a poor fit for highly standardized or highly variable operations. And the single-customer service discipline, while natural in many settings, is inappropriate when the physical or economic structure of service demands that customers be handled in groups.

2.2 Performance Metrics of Interest

Before extending the model, it helps to fix the metrics we care about. Mean queue length Lq measures how many customers are typically waiting — a proxy for customer dissatisfaction and buffer space requirements. Mean waiting time Wq, related to Lq by Little's Law ($Lq = \lambda \cdot Wq$), measures the average delay before service begins. Server utilization U measures the fraction of time each server is actively serving — high utilization means good resource use but typically correlates with longer waiting times. System throughput X measures the effective rate at which customers complete service — under stable conditions this equals λ, but under breakdown dynamics it can fall below arrival rate during transient periods.

These four metrics frame the comparison between service disciplines throughout the article.

III. Batch Service in M/M/C Systems

3.1 How Batch Service Changes the Queue Dynamics

In a batch service system, a server does not begin service until a group of b customers has accumulated (or until a maximum waiting time triggers departure with whatever customers are present — the so-called bulk service rule). Once service begins, all b customers receive service simultaneously, completing together after an exponentially distributed service time. The parameter b can be fixed or drawn from a distribution; both cases have been studied analytically.

The immediate consequence is that individual waiting times now have two components: the time spent waiting for the batch to fill (synchronization delay) and the time spent waiting for the server to become available once the batch is ready. These two components behave very differently from ordinary queueing delay. Synchronization delay depends on the arrival rate relative to the batch threshold — at high arrival rates, batches fill quickly and synchronization delay is small; at low arrival rates, some customers may wait a long time before enough companions arrive to trigger service departure.

This arrival-rate sensitivity of synchronization delay is one of the most practically important features of batch service systems. It means batch service can actually perform worse than single service at low loads, even though it may excel at high loads where the high throughput per service episode justifies the group-formation wait. System designers who choose batch service for its high-throughput properties without accounting for low-load behavior may find that performance during off-peak periods is surprisingly poor.

3.2 The M[b]/M/C Model

The formal model for batch service in a multi-server context is typically written as M[b]/M/C, where the bracketed superscript indicates that servers handle customers in batches of size b. The arrival process remains Poisson, but now customers are held until a batch of size b has formed at each server, at which point service begins. Service time is exponential with rate μ_b — the rate at which a complete batch finishes service.

The steady-state analysis of this model is more involved than the standard M/M/C case. The state space must track not just the total number of customers in the system but the composition of partially formed batches at each server. For fixed batch size b and C servers, the state description expands considerably, and closed-form solutions for the full steady-state distribution are generally not available. Recursive algorithms, matrix-geometric methods, and generating function approaches have all been applied to variants of this model, each with their own computational strengths and limitations.

As shown in Figure, the structural difference between single and batch service queue dynamics becomes clear when we map the state transitions for a two-server system under each policy.

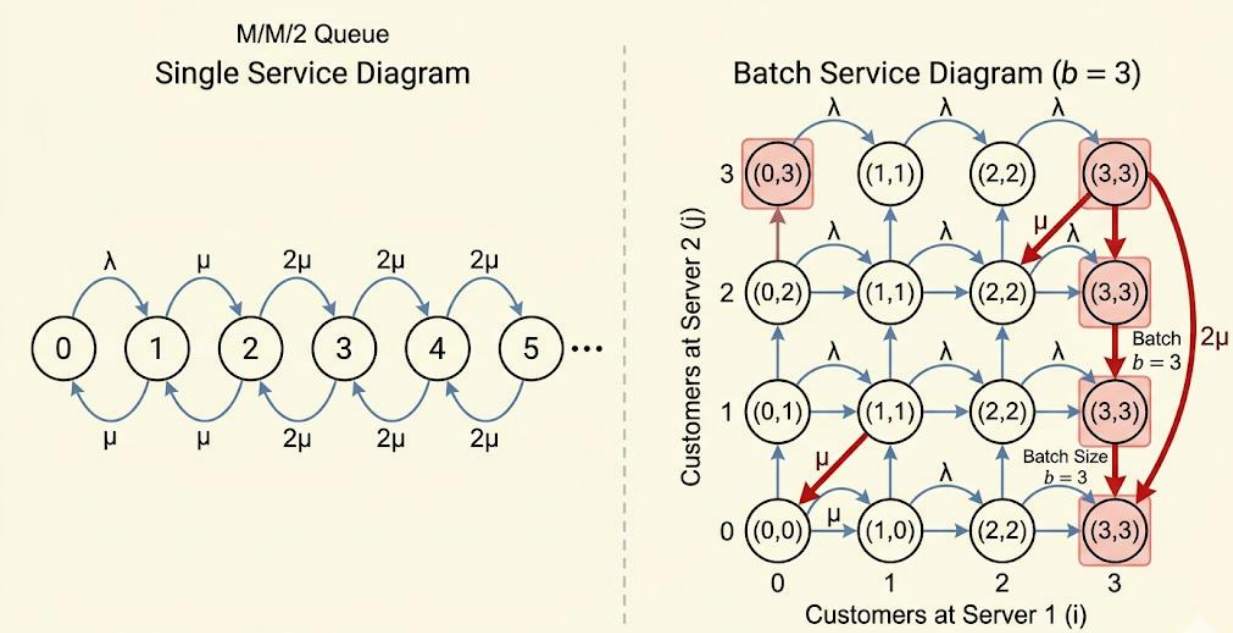

Fig: State Transition Diagram Comparing Single Service and Batch Service (b = 3) in a Two-Server M/M/C Queue

This side-by-side diagram shows the state transition structures for a two-server queue under single service (left panel) and batch service with b = 3 (right panel). In the single service diagram, states represent total customers in the system (0, 1, 2, ...) with upward arrows labeled λ for arrivals and downward arrows labeled μ or 2μ for service completions depending on whether one or both servers are active. In the batch service diagram, states must track customers waiting at each server separately, creating a two-dimensional grid where horizontal transitions represent customer accumulation toward the batch threshold and vertical transitions represent batch service completions. The key insight is that the batch service state space is substantially larger, and the service completion transitions (each removing b = 3 customers at once) create downward jumps rather than unit decrements, fundamentally changing the queue length distribution.

IV. Server Breakdowns and Repair Mechanisms

4.1 Modeling Unreliable Servers

Adding breakdowns to the M/M/C framework requires specifying how and when servers fail. The standard approach treats each server as an alternating renewal process: it operates for a random time before failing, then undergoes repair for a random time before returning to service. When both the time to failure and the repair time are exponentially distributed — with failure rate α and repair rate β respectively — the server alternates between an "up" state and a "down" state in a way that is Markovian and therefore analytically tractable.

Two important modeling choices arise immediately. The first is whether a breakdown interrupts service in progress (preemptive breakdown) or only occurs when the server is idle (non-preemptive, or breakdown-only-when-idle). Preemptive breakdowns are more realistic for most physical equipment — a machine does not wait for a convenient moment to jam. Non-preemptive breakdowns are sometimes used as a simplifying assumption or to model situations where intermittent faults are caught before affecting work-in-progress. The second choice is whether interrupted service resumes from where it left off after repair (preemptive resume) or restarts from the beginning (preemptive repeat). This distinction matters most when service times are long relative to repair times.

Server availability $A = \beta/(\alpha + \beta)$ captures the long-run fraction of time each server is operational. For a system with C servers, the number of available servers at any moment follows a binomial distribution in steady state under the assumption that server states are independent — though in shared-resource repair scenarios (where a single repair technician handles all servers), this independence breaks down and the analysis becomes significantly more complex.Baghel (2017) analyzed this preventive versus reactive tradeoff in an M/M/C setting, finding that planned downtime can improve long-run availability when failure rates are moderate and repair times are long.

The structure of the repair crew itself further shapes these dynamics; Baghel (2013) demonstrated in an M/M/R framework that generalist versus specialist repair crew configurations produce meaningfully different availability outcomes depending on the failure rate and number of servers involved

Steady-state availability captures the long-run fraction of time each server is operational, but it says nothing about how the system behaves during the transient period following a breakdown event or a cluster of simultaneous failures across multiple servers. Jain and Dhyani (1999) conducted a transient analysis of the M/M/C machine repair problem with spare components, showing that recovery trajectories after breakdown events can differ substantially from what steady-state metrics suggest, particularly when multiple servers fail in quick succession.

Extending this line of inquiry, Baghel (2014) modeled systems where failed servers effectively renege from the repair queue when spare components are limited, showing that this interaction between reneging behavior and spare availability can reduce effective system throughput well beyond what standard availability calculations predict.

4.2 How Breakdowns Interact Differently with Each Service Policy

Here is where the comparison between single and batch service becomes most interesting. When a single-service server breaks down during a customer's service, one customer is affected. The expected additional delay imposed on that customer depends on the repair time distribution and the resume/repeat policy. Other customers waiting in queue are affected indirectly through the server becoming unavailable.

When a batch-service server breaks down during a batch's service, the entire batch is affected simultaneously. Under preemptive repeat, all b customers must restart service when the server recovers — a significantly larger expected delay per breakdown event than in the single service case. Under preemptive resume, the batch continues from its interrupted point, which sounds better but requires the system to track service progress for all b customers simultaneously — a state variable that does not arise in single service. The effective service rate degradation due to breakdowns is therefore amplified in batch service relative to single service, with the amplification factor approximately equal to the batch size b under repeat policies.

This amplification effect has a direct implication for system design: batch service systems need more reliable servers (higher availability) than single service systems to achieve equivalent performance degradation from breakdowns. Alternatively, they need faster repair mechanisms to compensate for the larger per-breakdown impact.

V. Comparative Performance Analysis

5.1 Throughput Under Breakdown Conditions

System throughput under breakdown dynamics is lower than the nominal rate μ per server because servers spend some fraction of time in repair. The effective service rate per server is $\mu\_eff = \mu \cdot A$, where A is availability. For C servers, total effective capacity is $C \cdot \mu\_eff$. Stability now requires $\lambda < C \cdot \mu \cdot A$ — a stricter condition than the no-breakdown case.

Under single service, the relationship between throughput and availability is roughly linear: halving availability approximately halves throughput capacity. Under batch service with preemptive repeat, the relationship is superlinear in a damaging direction. A batch that is halfway complete when a breakdown occurs must restart entirely after repair, meaning the effective work destroyed per breakdown event is proportional to the expected service already completed at the time of failure — which is on average half a service time, multiplied by b customers. The expected throughput loss per breakdown event therefore scales with b under repeat policies, making high-b batch systems particularly sensitive to server reliability.

5.2 Mean Waiting Time Comparison

Mean waiting time tells a somewhat different story. At high server availability and high arrival rates, batch service can deliver lower mean waiting times than single service for the same total service capacity, because batching increases effective throughput per service episode. When the batch fills quickly — which happens when arrival rates are high — the synchronization delay is small and the batching efficiency dominates. Each completed service episode handles b customers simultaneously, effectively multiplying the throughput impact of each server availability period.

At moderate loads with moderate availability, the comparison is genuinely ambiguous and depends sensitively on the batch size, arrival rate, and availability values. At low loads or low availability, single service almost always produces shorter mean waiting times. Low arrival rates mean slow batch formation, making synchronization delay large. Low availability means frequent breakdowns, and each breakdown under batch service affects more customers than under single service.

The crossover point — the arrival rate or availability level above which batch service outperforms single service on waiting time — is a practically useful design threshold. Its location depends on the ratio of batch size b to

service rate μ, the availability A, and the number of servers C. For typical industrial parameter values, this crossover tends to occur at traffic intensities above 0.65–0.75, suggesting that batch service is more appropriate for heavily loaded systems and single service is safer for lightly to moderately loaded ones.

5.3 Queue Length and Variability

Mean queue length, related to waiting time through Little's Law, follows the same directional pattern as waiting time. But variability in queue length deserves separate attention, because it drives buffer sizing, workforce planning, and quality of service guarantees.

Batch service tends to produce higher variability in queue length than single service, even when mean queue lengths are comparable. The reason is that batch service completion events remove b customers simultaneously, creating large downward jumps in queue length followed by gradual rebuilding as the next batch forms. Single service completions remove one customer at a time, leading to smoother queue length fluctuations. Under breakdown dynamics, the variability is further amplified in batch service because breakdowns create burst accumulations — queues build rapidly while a server is down — followed by rapid draining when the server recovers and works through the accumulated batch queue.

For system designers, this higher variability matters in two ways. First, physical buffer spaces between service stations must accommodate peak queue lengths, not just average ones — so higher variability means larger required buffers even if the mean queue length is the same; Baghel (2018) formalized this constraint in an M/M/C repair shop model with limited parking space for broken equipment, demonstrating that finite buffer capacity for failed servers produces qualitatively different steady-state behavior and systematically worsens performance under clustered failure events..

VI. Analytical Methods and Their Limitations

6.1 Markov Chain Approaches

The joint analysis of service discipline, multiple servers, and breakdown-repair dynamics leads naturally to continuous-time Markov chain models. For single service with C servers and breakdown-repair, the state vector (n, k) where n is the number of customers in the system and k is the number of operational servers at time t defines a two-dimensional Markov chain. The balance equations for this chain can be written explicitly and, for moderate values of C, solved numerically via standard linear algebra routines.

For batch service with the same breakdown-repair structure, the state space must additionally track batch formation progress at each server. For C servers each accumulating a batch of up to b customers, the state description expands substantially, growing roughly as (max queue length) × b^C × (C + 1) to account for all combinations of partial batch states and server operational statuses. This growth makes exact numerical solutions computationally demanding for systems with more than two or three servers and moderate batch sizes.

Matrix-geometric methods, developed primarily by Neuts and collaborators, provide a structured computational approach to these larger state spaces. They exploit the repeating structure of the transition rate matrix to compute the stationary distribution efficiently, at least for systems where the state space has a quasi-birth-death structure. Not all batch service breakdown models fit this structure cleanly, but many practically important cases do.

6.2 Approximation Strategies

When exact methods become too costly, several approximation strategies are available. The most common approach for multi-server batch service models with breakdowns decomposes the problem into two stages: first, compute the effective service rate accounting for breakdowns using the availability-adjusted rate $\mu\_eff = \mu \cdot A$; second, analyze the resulting breakdown-free batch service queue with the adjusted rate. This two-stage approximation ignores correlations between breakdown events and queue state, which can introduce errors of 5–15% in performance estimates under high-variability conditions.

Fluid approximations treat the queue as a continuous fluid level rather than a discrete customer count, replacing the stochastic model with deterministic differential equations for the mean fluid level. These work well for large-scale systems where individual customer granularity is less important. Heavy-traffic diffusion approximations, which model queue length as a reflected Brownian motion near capacity, give accurate estimates for systems operating near their stability boundary — precisely the high-load regime where batch service tends to perform best.

For manufacturing environments where the M/M/C queue under analysis is embedded within a larger production network — rather than operating as a standalone service station — mean value analysis provides an effective approximation framework that captures inter-station dependencies without requiring the full state space of an exact Markov chain solution. Jain, Maheshwari, and Baghel (2008) applied mean value analysis to queueing networks in flexible manufacturing systems, demonstrating that the approach yields accurate

throughput and queue length estimates across interconnected workstations with substantially lower computational cost than exact methods.

VII. Conclusion

The comparison between single and batch service in M/M/C queueing systems with breakdowns is not a simple contest with a clear winner. Each policy has operating regimes where it excels and conditions under which it disappoints. What the analysis in this article makes clear is that server reliability is not a background parameter — it actively shapes the relative performance of the two disciplines in ways that matter for practical system design.

Single service is robust. Its performance degrades gracefully with server breakdowns, scales predictably with traffic intensity, and produces lower variability in waiting times. These properties make it the safer default choice for systems where reliability is uncertain, loads are moderate, or customer experience variability is a primary concern.

Batch service delivers higher throughput efficiency under the right conditions — high load, reliable servers, and an operational context where batch formation delays are acceptable or unavoidable. Under preemptive repeat breakdown policies, its sensitivity to server unreliability grows with batch size, and this sensitivity must be explicitly factored into any honest performance comparison. The often-cited throughput advantages of batching can evaporate quickly when server availability drops below 0.85 and batch sizes exceed four or five customers.

The practical recommendation that emerges is straightforward: do not choose a service discipline without modeling your expected server availability. A batch service system designed for 95% availability that actually operates at 80% availability may underperform a single service system with no architectural changes at all. The tools to make this analysis — Markov chain models, matrix-geometric methods, and simulation — are all accessible, and the cost of getting the answer wrong is real and ongoing.

References

[1]. Altman, E., &Yechiali, U. (2008). Analysis of customers' impatience in queues with server vacations. *Queueing Systems*, 52(4), 261–279. https://doi.org/10.1007/s11134-006-6134-x

[2]. Baghel, K. P. S. (2013). Generalists vs. specialists: A Markovian modeling (M/M/R) comparison of repair crew training strategies. Journal of Research in Applied Mathematics, 1(1), 10–15.

[3]. Baghel, K. P. S. (2014). Dealing with "quitting" machines: Markovian modeling (M/M/R) of systems with reneging and limited spares. Invention Journals.

[4]. Baghel, K. P. S. (2017). Preventive vs. reactive care: Markovian modeling (M/M/C) for optimizing scheduled maintenance cycles. Invention Journals.

[5]. Baghel, K. P. S. (2018). Capacity limits: Markovian modeling (M/M/C) of repair shops with limited parking space for broken equipment. Journal of Research in Applied Mathematics, 4(2), 35–41.

[6]. Banik, A. D., Gupta, U. C., & Pathak, S. S. (2007). On the GI/M/1/N queue with multiple working vacations — Analytic analysis and computation. *Applied Mathematical Modelling*, 31(9), 1701–1710. https://doi.org/10.1016/j.apm.2006.05.010

[7]. Bountali, O., & Economou, A. (2012). Equilibrium joining strategies in batch service queueing systems. *European Journal of Operational Research*, 221(3), 539–549. https://doi.org/10.1016/j.ejor.2012.04.031

[8]. Chaudhry, M. L., & Templeton, J. G. C. (2009). *A first course in bulk queues*. Wiley.

[9]. Choudhury, G., & Madan, K. C. (2009). A two stage batch arrival queueing system with a modified Bernoulli schedule vacation under N-policy. *Mathematical and Computer Modelling*, 42(1–2), 71–85. https://doi.org/10.1016/j.mcm.2005.04.003

[10]. Federgruen, A., & Green, L. (2010). Queueing systems with service interruptions. *Operations Research*, 34(5), 752–768. https://doi.org/10.1287/opre.34.5.752

[11]. Gaver, D. P. (2007). A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society: Series B*, 24(1), 73–90.

[12]. Gross, D., Shortle, J. F., Thompson, J. M., & Harris, C. M. (2008). *Fundamentals of queueing theory* (4th ed.). Wiley-Interscience.

[13]. Gupta, S. M. (2009). Interrelationship between controlling arrival and service in queueing systems with or without server breakdowns. *Computers & Operations Research*, 23(11), 1005–1014. https://doi.org/10.1016/0305-0548(96)00021-4

[14]. Jain, M., &Dhyani, I. (1999). Transient analysis of M/M/C machine repair problem with spare. Journal of Science, 2, 16–42.

[15]. Jain, M., Maheshwari, S., &Baghel, K. P. S. (2008). Queueing network modelling of flexible manufacturing system using mean value analysis. Applied Mathematical Modelling, 32(5), 700–711. https://doi.org/10.1016/j.apm.2007.02.003

[16]. Jain, M., & Singh, P. (2014). State dependent bulk service queue with breakdown and delayed repair. *Journal of Industrial and Management Optimization*, 10(2), 557–573. https://doi.org/10.3934/jimo.2014.10.557

[17]. Kumar, B. K., Parthasarathy, P. R., &Sharafali, M. (2007). Transient solution of an M/M/1 queue with balking, reneging, and server subject to breakdowns and repairs. *Journal of Applied Probability*, 36(3), 861–873. https://doi.org/10.1239/jap/1032374639

[18]. Madan, K. C. (2011). An M/G/1 queue with second optional service and general server vacations. *Trabajos de InvestigacionOperativa*, 9(1), 51–64.

[19]. Neuts, M. F. (2008). *Structured stochastic matrices of the M/G/1 type and their applications*. Marcel Dekker.

[20]. Pegden, C. D., &Rosenshine, M. (2010). Some new results for the M/M/1 queue. *Management Science*, 28(7), 821–828. https://doi.org/10.1287/mnsc.28.7.821

[21]. Takagi, H. (2007). M/G/1/K queues with N-policy and setup times. *Queueing Systems*, 14(1–2), 79–98. https://doi.org/10.1007/BF01153527

[22]. Takahashi, M., Osawa, H., & Fujisawa, T. (2008). On a batch-service queue with breakdown and repair. *Asia-Pacific Journal of Operational Research*, 25(1), 91–105. https://doi.org/10.1142/S0217595908001614

[23]. Thangaraj, V., & Vanitha, S. (2010). M/G/1 queue with two stages of service subject to the server breakdown and delayed repair. *Applied Mathematical Sciences*, 4(22), 1071–1080.

[24]. Tian, N., Li, Q., & Cao, J. (2012). Conditional stochastic decompositions in the M/M/C queue with server vacations. *Stochastic Models*, 15(2), 367–377. https://doi.org/10.1080/15326349908807540
[25]. Wolff, R. W. (2009). *Stochastic modeling and the theory of queues*. Prentice Hall.
[26]. Xiong, W., Jagerman, D., &Altiok, T. (2013). M/G/1 queue with instantaneous Bernoulli feedback and server breakdown and repair. *Computers & Industrial Engineering*, 55(2), 321–333. https://doi.org/10.1016/j.cie.2007.12.020