Ataques DoS Mitigados por Autoscaling: Uma Abordagem Analítica Transiente para Avaliar Sistemas Multicamadas

Francisco Airton Silva[1], **Iure Fé**[1], **Leonel Feitosa**[1], **Paulo Rego**[2] **e Bruno Nogueira** [3]

[1]Universidade Federal do Piauí (UFPI)

[2]Universidade Federal do Ceará (UFC)

[3]Universidade Federal de Alagoas (UFAL)

{faps,iure.fe,leonelfeitosa}@ufpi.edu.br

paulo@dc.ufc.br

bruno@ic.ufal.br

Resumo. *Ataques DoS (Denial of Service) ainda afetam sistemas de microserviços com múltiplas camadas, onde o congestionamento costuma aparecer primeiro na entrada do sistema. Para reduzir esse acúmulo, usa-se autoscaling para aumentar o paralelismo quando a ocupação cresce. Na prática, comparar configurações de autoscaling para preparação contra DoS é difícil porque os efeitos mais importantes aparecem ao longo do tempo (crescimento, pico e recuperação) e não apenas no regime estacionário. Este trabalho propõe uma abordagem analítica baseada em modelagem estocástica para estudar, de forma controlada e reprodutível, como parâmetros de autoscaling alteram a dinâmica temporal do congestionamento sob DoS. A avaliação foca na utilização da fila de entrada ao longo do tempo, permitindo observar pico, atraso até a reversão e tempo de retorno a níveis baixos.*

1. Introdução

Ataques DoS (Denial of Service) seguem afetando aplicações em nuvem construídas com microsserviços em múltiplas camadas, nas quais o tráfego é distribuído entre várias instâncias paralelas [Bremler-Barr et al. 2024]. Em geral, o evento se manifesta como uma elevação rápida de carga, com formação de um pico e, dependendo das condições de operação, uma fase posterior de recuperação. No entanto, esse padrão nem sempre indica um ataque: picos de demanda legítima também podem saturar canais digitais e produzir sintomas parecidos, como aconteceu no fim de 2025, quando o volume de apostas da Mega da Virada levou a instabilidades e filas nos sistemas da Caixa Econômica Federal [g1 2025]. Assim, independentemente da origem do pico, a característica temporal da sobrecarga influencia como o impacto se materializa ao longo da execução do sistema, o que chamamos de "janela de vulnerabilidade" [Kumari and Jain 2024].

Autoscaling é uma das respostas mais utilizadas nesse contexto, pois aumenta a capacidade do sistema ao criar novas réplicas quando a ocupação ultrapassa um limiar predefinido. Em geral, essa ação é reativa e depende de uma decisão baseada em métricas do próprio sistema, adicionando paralelismo para elevar a taxa agregada de atendimento [David 2021]. Entretanto, o resultado observado durante um ataque é fortemente influenciado pela parametrização, incluindo o limiar escolhido, a quantidade de réplicas adicionadas por evento, o intervalo de decisão e o tempo até as réplicas estarem prontas para atender. Essas escolhas determinam se a escala ocorre cedo o suficiente para conter o pico

ou se a sobrecarga continua se intensificando por tempo suficiente para degradar o serviço [Wang et al. 2022].

O problema que este trabalho investiga é que, antes de um incidente real, é difícil prever como um ataque DoS vai crescer, atingir um pico e depois recuar. Sem essa previsibilidade, também fica difícil definir como configurar o autoscaling: em que limiar ele deve reagir, quanto escalar a cada acionamento e quanto atraso pode ser tolerado até as novas instâncias começarem a atender. Pequenas diferenças nesses parâmetros podem mudar bastante o pico e o tempo em que o sistema permanece sobrecarregado, o que torna a escolha da configuração uma decisão incerta e difícil de justificar apenas com testes pontuais.

A literatura apresenta diversos estudos sobre arquiteturas em camadas sob DoS e o uso de autoscaling como mecanismo de reação, considerando diferentes padrões de carga e configurações [Wang et al. 2022, Bremler-Barr et al. 2017, Corrêa et al. 2019]. Existem métodos para analisar esses cenários, como testes de estresse, simulação e modelos analíticos; contudo, eles apresentam limitações quando se deseja comparar alternativas levando em conta a dinâmica transiente do evento. Testes e simulações costumam ser custosos para varrer amplos conjuntos de parâmetros, enquanto modelos analíticos frequentemente assumem regime estacionário ou simplificam o comportamento do autoscaling, especialmente atrasos de instanciação e políticas discretas. Assim, medições em pontos de equilíbrio são insuficientes, pois crescimento, pico e recuperação podem variar significativamente conforme a configuração, reforçando a necessidade de abordagens que permitam analisar esses efeitos de forma controlada antes da implantação ou de mudanças operacionais.

Este artigo explora a relação de ataques DoS a aplicações constituídas por microserviços e o mecanismo de autoscaling. Apresentamos uma abordagem analítica baseada em modelagem estocástica, que permite explorar sistematicamente cenários de configuração, variando parâmetros como o tempo até novas instâncias ficarem realmente disponíveis, o limiar de ocupação que aciona o scale-out e a quantidade de réplicas adicionadas por acionamento. A modelagem é construída com Redes de Petri Estocásticas (SPN), e o modelo proposto é validado com um simulador voltado a DoS/DDoS e autoscaling. Com isso, torna-se possível observar quando o congestionamento surge, qual a intensidade do pico, por quanto tempo o sistema permanece acima de níveis aceitáveis e em que condições a ocupação retorna a patamares baixos, apoiando escolhas de configuração orientadas a SLA.

O restante do artigo está organizado da seguinte forma: a Seção 2 define o cenário de ataque e os objetivos da análise; a Seção 3 sintetiza as principais abordagens da literatura sobre DoS e autoscaling; a Seção 4 descreve a abordagem de modelagem e as premissas adotadas para representar o ataque e a reação por escala; a Seção 5 avalia, no domínio transiente, o efeito de atrasos de instanciação, limiares de disparo e paralelismo adicionado; a Seção 6 apresenta a validação do modelo; e a Seção 7 resume os achados e elenca trabalhos futuros.

2. Delimitação do Problema

A Figura 1 apresenta o cenário-alvo assumido neste trabalho: uma aplicação em nuvem composta por microserviços organizados em camadas (FrontEnd, Processamento e Armazenamento), em que cada camada pode manter múltiplas instâncias/réplicas para atender

requisições em paralelo [Poojara et al. 2022]. Esse tipo de organização é comum porque facilita evolução e implantação independente, mas também cria um ponto sensível: em geral, todo o tráfego converge para uma porta de entrada (por exemplo, um gateway/Nginx), onde um ataque DoS se concentra e transforma rapidamente o aumento de chegadas em acúmulo de requisições na fila de entrada.

Em seguida, a pressão se propaga para as camadas seguintes, elevando tempos de espera e a chance de descarte, além de provocar instabilidade no atendimento quando a capacidade de processamento não acompanha a demanda. Assim, alguma ação de mitigação precisa ocorrer já no ponto de entrada ou no mecanismo de provisão de recursos do serviço. Entre as alternativas, o autoscaling se destaca por ser amplamente adotado em ambientes de nuvem: ele aumenta a capacidade efetiva do sistema ao criar novas réplicas, buscando reduzir o congestionamento e manter o serviço dentro de limites de qualidade, mesmo durante a fase inicial do ataque.

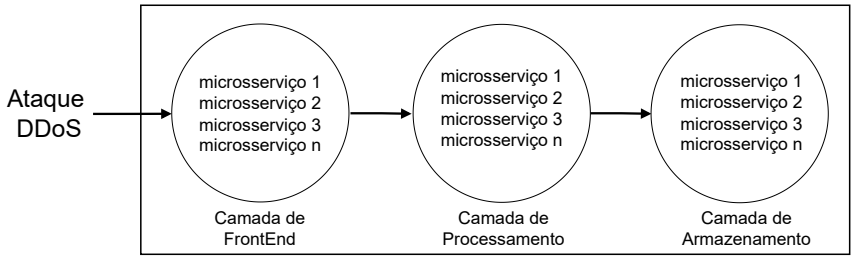


Figura 1. Típica arquitetura em camadas sob ataque DoS na camada de entrada.

A Figura 2 representa uma situação comum na área [Khan et al. 2021, Xie et al. 2024, Huang et al. 2025]. O autoscaling é representado como uma ação reativa baseada na ocupação da fila de entrada: quando o nível de ocupação ultrapassa um limiar x%, o mecanismo de controle dispara a criação de novas réplicas do microserviço responsável pelo atendimento naquela camada, injetando capacidade paralela adicional (por exemplo, três réplicas) para aumentar a taxa agregada de serviço. Na prática, isso "despressuriza" a fila ao distribuir as requisições entre mais instâncias, reduzindo a permanência em congestionamento e a probabilidade de colapso durante a sobrecarga. Ao mesmo tempo, o resultado depende da parametrização: o limiar escolhido, a quantidade de réplicas adicionadas por evento de escala, o intervalo de decisão e o tempo até essas réplicas estarem prontas para atender, porque essas escolhas determinam se o scale-out ocorre cedo o bastante para conter o pico ou se a fila continua crescendo por tempo suficiente para degradar o serviço.

A Figura 2 ilustra um cenário recorrente na literatura, no qual o autoscaling atua como um mecanismo reativo baseado na ocupação da fila de entrada [Khan et al. 2021, Xie et al. 2024, Huang et al. 2025]. Neste trabalho, ataques DoS são modelados como eventos de sobrecarga adversarial, caracterizados por um crescimento rápido e intenso da taxa de chegada, sem assumir mecanismos explícitos de detecção ou filtragem do tráfego malicioso. Assim, o autoscaling reage apenas ao efeito do ataque (a saturação da fila) criando novas réplicas quando a ocupação ultrapassa um limiar x%, com o objetivo de injetar capacidade paralela adicional e sustentar a taxa de atendimento. Embora essa reação seja semelhante à adotada em picos de demanda legítima, a diferença está na dinâmica do evento: em ataques DoS, o crescimento abrupto, o pico mais pronunciado e a duração da sobrecarga ampliam a janela de vulnerabilidade do sistema, tornando críticos parâmetros

como limiar de disparo, atraso de instanciação e paralelismo adicionado.

Definir previamente quais valores são "ideais" é difícil, já que o comportamento depende do estado atual do sistema e do padrão de carga/ataque; por isso, torna-se necessário antecipar como o sistema tende a reagir sob diferentes configurações antes da implantação ou de mudanças operacionais. Nesse ponto, modelos analíticos podem ser usados para prever tendências de comportamento e comparar alternativas ainda em fases iniciais de design, apoiando a escolha de parâmetros de autoscaling com base em cenários controlados.

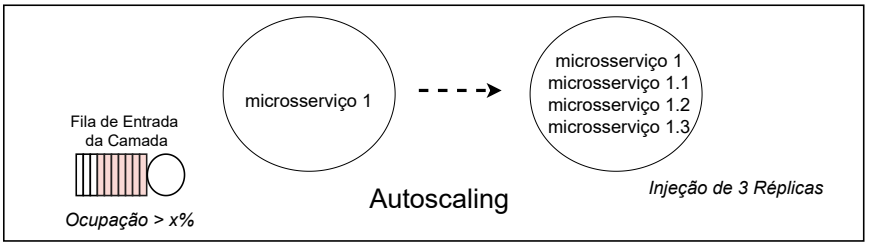


Figura 2. Autoscaling acionado quando a fila de entrada ultrapassa o limiar.

3. Trabalhos Relacionados

Esta seção revisa trabalhos relacionados que investigam ataques de negação de serviço em ambientes elásticos, com ênfase em autoscaling e em abordagens de modelagem analítica ou estocástica. A literatura apresenta diferentes perspectivas, variando desde experimentos em plataformas de nuvem até modelos formais que buscam caracterizar o impacto temporal dos ataques sobre filas e recursos computacionais. A Tabela 1 compara seis trabalhos levantados e o nosso estudo por quatro dimensões: se o estudo aborda relação com o trabalho (autoscaling e/ou modelagem analítica), tipo de ataque, perfil temporal do ataque e método de avaliação. A organização permite ver quais estudos tratam elasticidade/autoscaling [Wang et al. 2022, Bremler-Barr et al. 2017, Corrêa et al. 2019] e quais se apoiam principalmente em modelagem estocástica/analítica [Khan et al. 2021, Mutar et al. 2024], além de cenários baseados em simulação de rede [Wei et al. 2017], e posicionar o nosso trabalho nesse conjunto.

Tabela 1. Trabalhos sobre DoS que exploraram autoscaling e/ou modelagem.

Ref.	Relação	Tipo de ataque	Perfil temporal do ataque	Método de avaliação
[Wang et al. 2022]	Autoscaling	DDoS assimétrico (consumo/custo)	Sustentado por etapas (variação em níveis)	Experimentos em plataforma serverless
[Bremler-Barr et al. 2017]	Autoscaling	Abuso de autoscaling (yo-yo / EDoS)	Pulsado on/off (oscilante)	Experimentos em nuvem + análise do comportamento
[Corrêa et al. 2019]	Autoscaling	ADDoS low-rate + HTTP GET flood	Prolongado (low-rate) e/ou abrupto (flood)	Experimentos em ambiente de nuvem
[Wei et al. 2017]	Modelo (simulação de rede/fila)	Flooding (ex.: SYN/RST)	Aumento de intensidade (variação de taxa)	Simulação (NS2)
[Khan et al. 2021]	Modelo analítico (Markov)	Flooding (DoS/DDoS)	Probabilístico no tempo (cadeia de Markov)	Modelo estocástico + simulação
[Mutar et al. 2024]	Modelo analítico (CTMC)	HTTP-GET flood	Cenários com variação de taxa ao longo do tempo	Modelo analítico (CTMC)
Este Trabalho	Autoscaling + Modelo (SPN)	Ambos DoS ou DDoS	Abrupto	SPN com análise transiente e varredura paramétrica

Relação com o trabalho (autoscaling/modelo analítico). Os trabalhos se dividem entre estudos centrados em autoscaling e estudos centrados em modelagem.

[Wang et al. 2022], [Bremler-Barr et al. 2017] e [Corrêa et al. 2019] tratam autoscaling como mecanismo do sistema (ou como alvo do ataque/defesa). [Khan et al. 2021] e [Mutar et al. 2024] usam modelagem analítica/estocástica (Markov e CTMC) para caracterizar ataque e impacto. [Wei et al. 2017] fica no bloco de modelagem via simulação de rede e mecanismos de fila. No nosso trabalho, há os dois: autoscaling como resposta e SPN.

Tipo de ataque. Há diferenças no vetor. [Corrêa et al. 2019] inclui ADDoS low-rate e HTTP GET flood. [Mutar et al. 2024] usa HTTP-GET flood em LTE-M. [Wei et al. 2017] e [Khan et al. 2021] tratam flooding em rede. [Bremler-Barr et al. 2017] discute ataques que exploram o mecanismo de autoscaling específico yo-yo/EDoS. [Wang et al. 2022] trata um ataque assimétrico em serverless. No nosso trabalho, o ataque base é Flooding DoS.

Perfil temporal do ataque. O perfil temporal define o que precisa ser observado. [Bremler-Barr et al. 2017] usa on/off para induzir oscilações no autoscaling. [Wei et al. 2017] varia a intensidade para comparar efeitos em filas. [Khan et al. 2021] usa probabilidade de ataque variando no tempo (Markov). [Wang et al. 2022] usa carga em níveis/etapas. [Mutar et al. 2024] considera cenários com taxa variável ao longo do tempo. No nosso estudo, o perfil é abrupto para medir pico e tempo de recuperação na análise transiente.

Método de avaliação. Os métodos também se separam. [Wang et al. 2022], [Bremler-Barr et al. 2017] e [Corrêa et al. 2019] usam experimentos em ambientes de nuvem/plataforma. [Wei et al. 2017] usa simulação em NS2. [Khan et al. 2021] usa modelos estocásticos (Markov) com avaliação em simulação. [Mutar et al. 2024] usa CTMC com solução analítica. No nosso trabalho, a avaliação é por SPN com análise transiente e estacionária.

Principais diferenças do nosso trabalho em relação aos demais. O nosso trabalho concentra-se no comportamento transiente da fila de entrada sob DDoS e combina, no mesmo modelo SPN, ataque (flood DoS), paralelismo, limiar de gatilho e atraso de instanciação. Em [Bremler-Barr et al. 2017] e [Wang et al. 2022], a ênfase está em mostrar como o autoscaling pode ser explorado (oscilação/custo) em cenários de nuvem ou serverless; no nosso caso, o foco é comparar como parâmetros e decisões alteram pico e recuperação da fila. Em [Corrêa et al. 2019], a discussão combina mecanismos (seletividade e autoscaling) via testbed; no nosso estudo, a comparação é orientada por varredura paramétrica em SPN. Em [Wei et al. 2017, Khan et al. 2021], o alvo é filas sem autoscaling; em [Mutar et al. 2024], o foco é inferência por estados, não a resposta do autoscaling.

4. Modelo SPN Proposto

Em redes de Petri estocásticas, o comportamento do sistema é descrito por um grafo bipartido formado por lugares (círculos) e transições (barras), conectados por arcos que definem as relações de entrada e saída entre esses elementos [Maciel 2023]. Na Figura 3, os lugares representam estados/recursos do sistema, as transições representam eventos que mudam o estado, e a dinâmica ocorre pelo movimento de marcações (tokens) entre lugares quando uma transição dispara, respeitando as condições impostas pelos lugares e arcos (isto é, disponibilidade de marcações e limites de capacidade).

Na Figura 3, uma camada n da arquitetura é representada como uma SPN com-

posta por uma Fila de Espera e uma Fila de Processamento. As requisições chegam e são armazenadas na fila de espera (lugares associados à entrada), onde permanecem por um tempo controlado pela transição temporizada TE2; o disparo de TE2 é condicionado pela disponibilidade de espaço na fila, modelada pela capacidade Q1. Após TE2, a requisição só avança para a fila de processamento se houver capacidade disponível no lugar P4, cuja capacidade inicial é dada pela marcação C1. Uma vez na fila de processamento, o atendimento é modelado pela transição temporizada TE3, que representa a execução do processamento; ao disparar TE3, a requisição deixa o processamento e segue para a camada seguinte, mantendo a lógica de fluxo em camadas que a arquitetura assume.

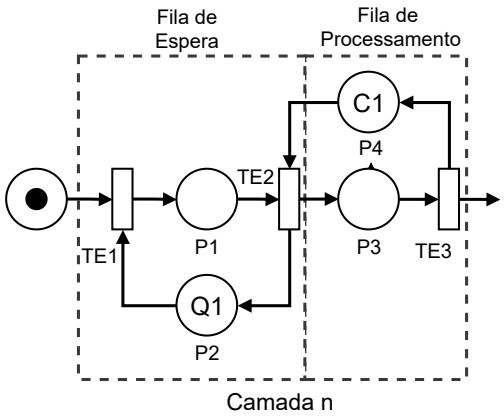


Figura 3. Visão SPN de uma única camada.

A Figura 4 apresenta o modelo SPN proposto. O modelo foi desenvolvido e executado na ferramenta Mercury [Silva et al. 2015], e segue uma estrutura clássica em pipeline por camadas, separando explicitamente a Geração de Carga (à esquerda) do Sistema Alvo (à direita), que é dividido em três camadas: FrontEnd, Processamento e Armazenamento. Nesse arranjo, as requisições são geradas, em seguida, percorrem as camadas do sistema em sequência, permitindo observar como o fluxo se propaga, onde ocorre acúmulo de tokens (fila) e como a capacidade de serviço influencia o comportamento ao longo do tempo. Vale ressaltar que o modelo é adaptável, como por exemplo, sendo possível a adição e remoção de camadas ou filas internas, e suporta configuração de vários parâmetros relacionados a capacidades (lugares) e tempos (transições).

Na Geração de Carga, há dois regimes de geração para possibilitar a análise transiente com precisão. Inicialmente, a transição temporizada `TE1` representa uma fase de operação com baixa carga, com maior tempo entre disparos (isto é, menor taxa de chegadas). No início, `TE1` começa habilitada pois há um arco inibidor para o lugar `P2`, se mantendo habilitada enquanto não tiver token em `P2`. Após um intervalo fixo, controlado pela transição determinística `TD1`, o estado de geração muda, colocando um token em `P2` e habilitando a transição temporizada `TE2`. As demais transições temporizadas possuem distribuição exponencial. Como `TE2` está conectada por um arco bidirecional com `P2`, ela permanece continuamente habilitada a partir desse ponto, passando a gerar requisições com menor tempo entre disparos (maior taxa), caracterizando a carga imposta durante o ataque DoS. Aqui é importante salientar que o modelo abstrai se o ataque é do tipo distribuído (DDoS) ou não (DoS) pois não consideramos de onde partem as requisições e sim que elas chegam em um único ponto do sistema.

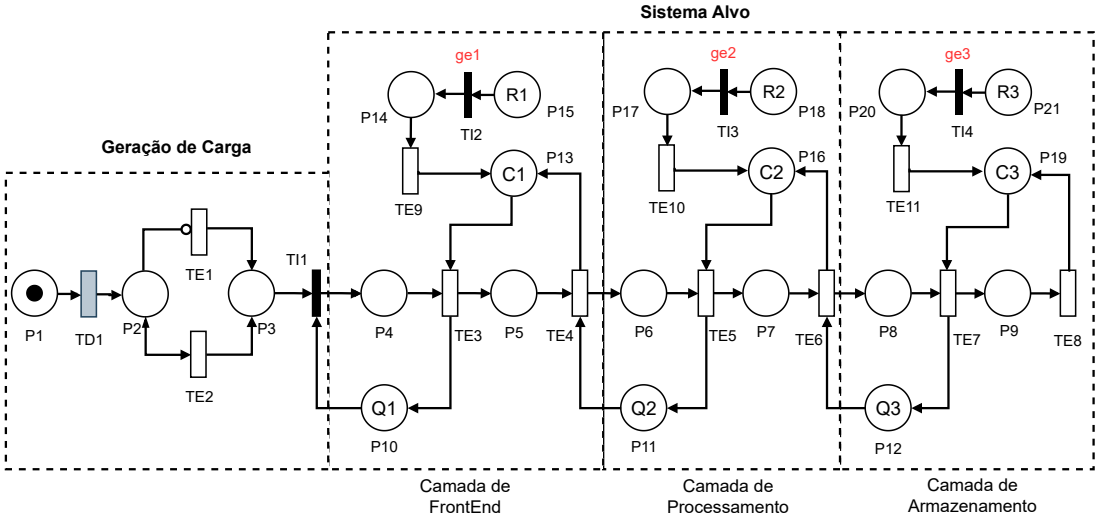


Figura 4. Modelo SPN proposto.

Quando a transição imediata `TI1` dispara, a requisição sai do gerador e entra na primeira camada do sistema (FrontEnd). Cada camada do pipeline mantém duas estruturas: uma fila de espera (entrada) e uma fila de processamento (serviço), com transições temporizadas que representam o avanço entre essas etapas e o envio para a próxima camada (por exemplo, no FrontEnd aparecem as transições `TE3` e `TE4`, e de forma análoga nas camadas seguintes). O diferencial em relação ao modelo básico de filas (explicado anteriormente) é o submodelo superior de scale-out presente em cada camada. No FrontEnd, a condição de guarda `ge1` controla a transição imediata `TI2`, que é habilitada quando a utilização observada na fila de espera `P4` ultrapassa um limiar.

A guarda ge1 = $(\#P4 \times 100 > Q1 \times percent)$ define a condição de disparo do scale-out na camada. Nessa expressão, `#P4` é o número de marcações em `P4`, isto é, a quantidade de requisições atualmente na fila de espera dessa camada; `Q1` é a capacidade máxima dessa fila; e $percent$ é o limiar configurado. Ao multiplicar `#P4` por 100 e comparar com $Q1 \times percent$, a guarda verifica se a ocupação da fila, em termos percentuais, ultrapassou o limiar. Quando essa desigualdade se torna verdadeira, `ge1` habilita a transição imediata associada ao autoscaling, permitindo iniciar a instanciação de novas réplicas para aumentar a capacidade de atendimento e reduzir a pressão sobre a fila. As outras condições de guarda seguem a mesma ideia.

Quando `TI2` dispara, uma quantidade de réplicas representada pela marcação `R1` é convocada, e a disponibilização dessas novas instâncias ocorre após o atraso de instanciação modelado por `TE9`; ao final desse atraso, a capacidade efetiva da camada aumenta ao reforçar a marcação associada à capacidade (inicialmente `C1`). O mesmo padrão se repete nas demais camadas, com ge2/TI3/R2/TE10/C2 no Processamento e ge3/TI4/R3/TE11/C3 no Armazenamento, sempre condicionando o scale-out ao nível de utilização da fila de espera da própria camada e alterando o paralelismo disponível para reduzir acúmulo e sustentar o escoamento do pipeline.

A Equação 1 define a métrica principal utilizada neste estudo, U, que mede o nível de utilização da fila de espera na camada de entrada do sistema. Nessa formulação, $E\{\#P4\}$ representa o número esperado de marcações no lugar `P4`, isto é, a quantidade

média de requisições acumuladas na fila de entrada, enquanto `Q1` é a capacidade máxima dessa fila. Ao normalizar o conteúdo médio da fila pela sua capacidade e multiplicar por 100, obtém-se a utilização em porcentagem, permitindo acompanhar ao longo do tempo o quão próxima a camada de entrada está da saturação e, assim, avaliar o efeito das estratégias de autoscaling na redução do congestionamento. Vale ressaltar que observamos a fila de entrada no sistema pois é a última fila a chegar a 100% de utilização, as demais camadas neste ponto já estão exauridas.

$$U = \left(\frac{E\{\#P4\}}{Q1} \right) \times 100 \tag{1}$$

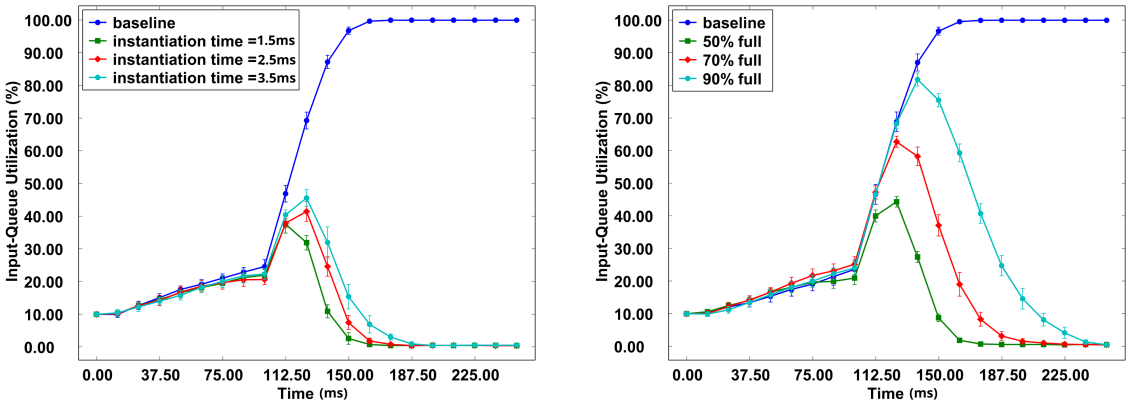
5. Estudos de Caso

A Tabela 2 resume os principais parâmetros adotados no modelo. Ela inclui as transições temporizadas associadas à geração de requisições ($TE1$ e $TE2$), aos atrasos internos de cada camada (alocação e processamento), ao tempo de instanciação de novas réplicas e ao instante determinístico em que o regime de carga muda ($TD1$). Além disso, a tabela apresenta as capacidades das filas de espera ($Q1$–$Q3$), a capacidade de processamento paralelo em cada camada ($C1$–$C3$), a quantidade de réplicas adicionadas em eventos de scale-out ($R1$–$R3$) e o limiar percent que aciona o autoscaling. Todas as execuções foram feitas no modo simulação transiente ou estácionária com erro de 5%. As transições $TE9$, $TE10$, $TE11$ executam no modo paralelo (infinite server), as demais transições temporizadas são single server (sequencial) [Mura 2015]. As análises transientes foram observadas na janela de 250ms.

Tabela 2. Parâmetros do modelo.

Categoria	Parâmetro(s)	Descrição	Valor
Transições	TE1	Tempo médio de geração antes do ataque	2,0 ms
	TE2	Tempo médio de geração durante do ataque	0,3 ms
	TE3, TE5, TE7	Tempo médio de alocação de requisições	1,0 ms
	TE4, TE6, TE8	Tempo médio de processamento	1,2 ms
	TE9, TE10, TE11	Tempo médio de instanciação de réplicas	1,0 ms
	TD1	Tempo médio para iniciar ataque	100,0 ms
Capacidades	Q1, Q2, Q3	Capacidade inicial de fila	100
	C1, C2, C3	Capacidade inicial de microsserviços	2
	R1, R2, R3	Novos microsserviços a serem instanciados	10
	percent	Percentual de gatilho do autoscaling	50%

A Figura 5 reúne duas visões transientes da métrica `U` (utilização da fila de entrada) ao longo do tempo, comparando sempre um baseline (sem autoscaling) com configurações que ativam scale-out. Em ambos os gráficos, o sistema inicia em um regime de baixa carga, com utilização em torno de 10%, e por volta de $t \approx 100ms$ ocorre a mudança de regime que representa o ataque, elevando abruptamente a pressão de chegada. Nessa condição, o baseline tende a caminhar para saturação (100%), enquanto as curvas com autoscaling apresentam um pico durante a fase inicial do ataque e, após a disponibilização de capacidade adicional, passam a reduzir a utilização até níveis próximos de zero, evidenciando o efeito do scale-out na contenção do congestionamento.



(a) Variação do tempo de instanciação. **(b) Variação do gatilho de utilização.**

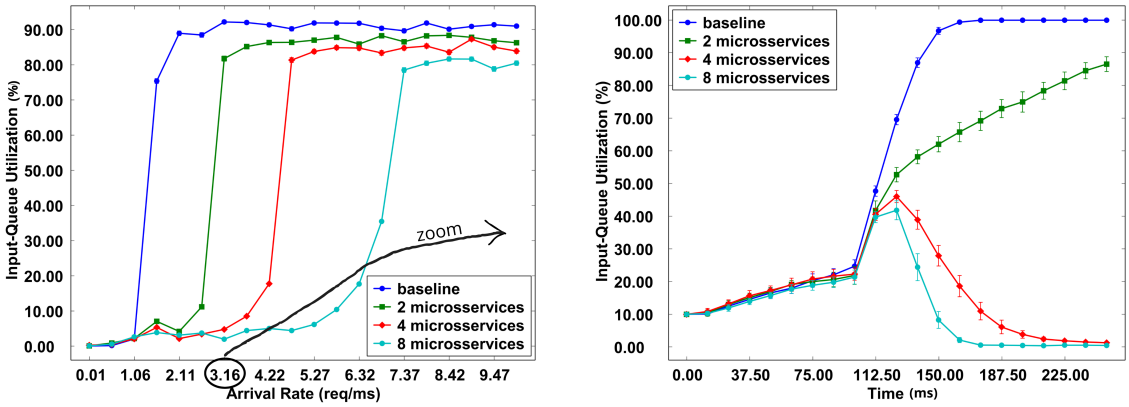
Figura 5. Instanciação e limiar de utilização x sobrecarga do sistema.

Na Figura 5a, o parâmetro variado é o tempo de instanciação das novas réplicas. Mantendo o mesmo padrão de ataque, observa-se que o autoscaling consegue reverter o crescimento da fila após o pico, porém o momento em que a curva vira e a velocidade de queda dependem desse atraso: quanto menor o tempo de instanciação, mais cedo a capacidade adicional entra em operação e mais rapidamente a utilização da fila converge para valores baixos. Já os tempos de instanciação maiores deslocam a recuperação para mais tarde, prolongando a permanência em níveis elevados de utilização antes que a fila seja escoada.

Na Figura 5b, o parâmetro variado é o limiar de utilização que aciona o gatilho de scale-out (porcentagem "full"). Limiar menor (por exemplo, 50%) dispara o autoscaling mais cedo, limitando o crescimento do pico e antecipando a redução da utilização, enquanto limiares maiores (70% e 90%) postergam o acionamento e permitem que a fila acumule mais antes da reação, resultando em picos mais altos e recuperação mais tardia. Em termos operacionais, o gráfico sugere o trade-off esperado: gatilhos mais precoces tendem a reduzir a janela de congestionamento, mas podem manter o sistema escalado por mais tempo, aumentando o período de uso de recursos e custos.

A Figura 6 explora o efeito do paralelismo adicionado pelo autoscaler (2, 4 ou 8 réplicas em cada camada) na métrica `U`. A Figura 6a resume uma visão estacionária, isto é, o valor de `U` após o sistema atingir regime para cada nível de carga (entendido como diferentes intensidades de ataque). A Figura 6b mostra a visão transiente para um caso específico de carga, permitindo enxergar o pico, o instante em que a fila começa a escoar e a tendência de recuperação. Em conjunto, os dois gráficos mostram que aumentar o número de réplicas desloca o ponto em que a fila passa a permanecer alta, mas esse ganho tem limite quando a carga é muito elevada.

Na Figura 6a , o eixo x representa a taxa de chegada (req/ms), e cada curva mostra a utilização estacionária resultante ao configurar o autoscaling. Para cargas baixas (por exemplo, em torno de 1,06 req/ms), o sistema não é pressionado a ponto de manter a fila ocupada: `U` fica baixo para todas as configurações. À medida que a carga aumenta, surge um ponto de virada em que o autoscaling com poucas réplicas deixa de ser suficiente e `U` passa a se manter em níveis altos. No ponto 3,16 req/ms, adicionar 4 ou 8 microsserviços mantém a utilização em patamares baixos, enquanto adicionar apenas 2 leva o sistema para uma região próxima de 80%. Para cargas extremas (por exemplo,



(a) Visão estacionária. **(b) Visão transiente.**

Figura 6. Quantidade de microsserviços adicionais x sobrecarga do sistema.

9,47 req/ms), todas as curvas ficam em níveis elevados (80–90%), indicando que, nesse regime, mesmo aumentar a quantidade de réplicas por acionamento não é suficiente para evitar alta ocupação em longo prazo, e seria necessário rever capacidade base, política de escala, ou outras formas de mitigação.

Na Figura 6b, fixando o cenário de carga (um "zoom" do caso em torno de 3,16 req/ms), a visão transiente confirma o que a análise estacionária sugere. Antes do ataque, a utilização permanece em um patamar baixo; após a mudança de regime, $\mathtt{U}$ cresce rapidamente e atinge um pico. Quando o autoscaling adiciona mais paralelismo, as curvas com 4 e 8 microsserviços passam a cair e convergem para valores próximos de zero, indicando recuperação da fila após a entrada das réplicas. Em contraste, com 2 microsserviços, a curva não recupera da mesma forma e se estabiliza em níveis altos, coerente com o resultado estacionário. Assim, os dois gráficos se complementam: enquanto o (a) ajuda a identificar quanto de paralelismo é necessário para cada intensidade de ataque em regime, o (b) mostra como essa diferença aparece no tempo (pico, atraso e recuperação).

Os estudos apresentados nesta seção mostram que o comportamento do congestionamento sob DoS é fortemente condicionado por poucos parâmetros centrais do autoscaling e, principalmente, pela forma como seus efeitos se desenrolam no tempo. Ao observar a utilização da fila de entrada, foi possível identificar o pico, o instante de reversão e o tempo de recuperação, evidenciando que atrasos de instanciação e limiares de disparo deslocam diretamente a janela de vulnerabilidade, enquanto o paralelismo adicionado atua como fator de suporte com retorno mais visível em cargas moderadas e limites sob intensidades elevadas. Em conjunto, os resultados indicam que o modelo SPN é capaz de capturar e comparar, de modo controlado, os principais parâmetros do autoscaling (limiar, atraso de provisão e tamanho do scale-out), permitindo analisar cenários "what-if" e apoiar a seleção de configurações mais aderentes a metas operacionais e de SLA antes da implantação ou de mudanças no sistema.

5.1. Análise de Sensibilidade

O planejamento de experimentos (Design of Experiments – DoE) [Eriksson et al. 2000] foi utilizado para investigar, de forma controlada, como escolhas de configuração do autoscaling influenciam a resposta do sistema medida por $U1$, evitando comparações ad hoc e permitindo isolar causas prováveis das variações observadas. Para isso, definiu-se

um experimento fatorial com três fatores e dois níveis por fator (baixo e alto), de modo que cada parâmetro fosse testado em condições contrastantes: time_injection (1,5 e 3,5), new_pods (2 e 8) e percent (50 e 90), conforme sumarizado na Tabela 3. A partir desses níveis, foram geradas todas as 8 combinações possíveis (2^3), e cada combinação foi executada para obter o valor correspondente de $U1$, apresentado na Tabela 4. Com esses resultados, o gráfico de Pareto é empregado para ranquear os efeitos principais (cada fator isoladamente) e as interações entre fatores, evidenciando quais parâmetros explicam a maior parcela da variação de $U1$ e, portanto, devem ser priorizados na calibração do autoscaling; além disso, as análises de interação complementam o Pareto ao indicar quando o efeito de um fator depende do nível de outro, o que orienta ajustes mais coerentes do limiar de disparo, do atraso de provisão e do tamanho do scale-out.

Tabela 3. Fatores e níveis.

Fator	Low Setting	High Setting
time_injection	1,5	3,5
new_pods	2,0	8,0
percent	50,0	90,0

Tabela 4. Combinações DoE.

time_injection	new_pods	percent	U
1,5	2	50,0	16,75
1,5	2	90,0	22,65
1,5	8	50,0	11,68
1,5	8	90,0	21,05
3,5	2	50,0	22,41
3,5	2	90,0	22,79
3,5	8	50,0	18,75
3,5	8	90,0	24,66

A Figura 7 apresenta o gráfico de Pareto [Santos et al. 2024], com o nível de impacto (Effect) dos fatores e de suas interações. Nesse contexto, percent representa o limiar/percentual usado como gatilho na regra de decisão (isto é, o quanto precisa ocorrer para acionar a ação), time_injection corresponde ao instante/tempo em que a injeção (por exemplo, de réplicas/pods) é aplicada ao longo da execução, e new_pods indica a quantidade de novas réplicas/pods adicionadas quando a ação é acionada.

Observa-se que percent é o parâmetro mais influente (efeito próximo de 7,8), seguido por time_injection (cerca de 4,2), enquanto a interação time_injection × percent ainda mantém contribuição perceptível (por volta de 2,2). Já new_pods e new_pods × percent aparecem com efeitos menores (em torno de 1,0–1,2) e time_injection × new_pods é o termo menos expressivo (aprox. 0,6), o que direciona o ajuste do sistema: priorizar a calibração de percent (com faixas e níveis bem definidos), em seguida variar time_injection para comparar diferentes instantes de injeção e escolhê-lo condicionado a cada faixa de percent, deixando new_pods como ajuste fino após as escolhas principais, dado o baixo retorno em investir esforço específico na combinação time_injection × new_pods.

6. Validação

Realizamos uma validação do modelo usando um simulador específico para a questão de autoscaling chamado DDoS-AS-Sim[1] escrito em Java e com as mesmas parametrizações do modelo SPN. Para a validação usamos uma versão mais simples do modelo, com apenas a camada de entrada (Frontend). Para evidenciar a adaptabilidade do modelo variamos alguns parâmetros em relação ao estudo de caso anterior. A Tabela 5 apresenta os

[1]https://gitlab.com/pasid3/ddos-autoscaling-simulator

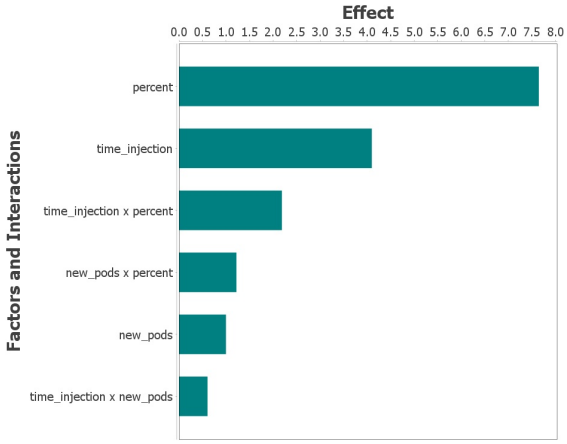


Figura 7. Nível de impacto dos fatores.

parâmetros usados nesta etapa. Por restrição de espaço não apresentaremos o submodelo, basta verificar no modelo principal apresentado na Seção 4.

Tabela 5. Parâmetros do submodelo (apenas com a primeira camada).

Categoria	Parâmetro(s)	Descrição	Valor
Transições	TE1	Tempo médio de geração antes do ataque	100,0 ms
	TE2	Tempo médio de geração durante do ataque	2,0 ms
	TE3	Tempo médio de alocação de requisições	0,1 ms
	TE4	Tempo médio de processamento	50,0 ms
	TE9	Tempo médio de instanciação de réplicas	5,0 ms
	TD1	Tempo médio para iniciar ataque	200,0 ms
Capacidades	Q1	Capacidade inicial de fila	100
	C1	Capacidade inicial de microsserviços	2
	R1	Novos microsserviços a serem instanciados	30
	percent	Percentual de gatilho do autoscaling	50%

Testamos uma configuração sem autoscaling (baseline) e uma com adição de 30 microsserviços em um tempo de análise transiente de 1000ms. A Figura 8 apresenta o resultado da validação. O comportamento dinâmico global é consistente: ambos capturam o mesmo padrão de resposta à mudança de regime (início do ataque/elevação de carga), o surgimento de um pico de utilização e a subsequente queda após o efeito do provisionamento. A validação aqui é sustentada pelo comportamento estacionário: observa-se que, no baseline, tanto o modelo quanto o simulador convergem para saturação, estabilizando a utilização próxima de 100%, indicando que a capacidade disponível é insuficiente sob ataque e a fila permanece cheia. Já no cenário com +30 microserviços, ambos convergem para utilização próxima de 0%, refletindo que o aumento de capacidade absorve a carga e esvazia a fila de entrada. Assim, existe a concordância nos regimes estacionários fornecendo a evidência de que o modelo representa adequadamente o sistema, sendo, portanto, validado pelo simulador para os objetivos do estudo.

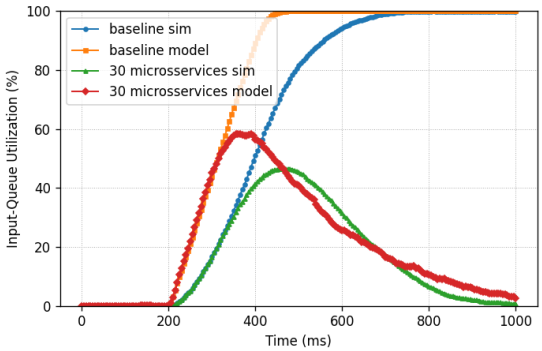


Figura 8. Validação do modelo usando o simulador DDOS-AS-Sim.

7. Conclusão

Este trabalho apresentou uma abordagem analítica para estudar o comportamento transiente do autoscaling sob ataques DoS, observando a evolução da ocupação da fila de entrada ao longo do tempo. Os resultados mostram que o atraso entre o acionamento do scale-out e a disponibilidade real de novas instâncias influencia diretamente o pico e o tempo de recuperação: atrasos menores antecipam a queda da ocupação, enquanto atrasos maiores prolongam a permanência em níveis altos. O limiar de ocupação usado como gatilho também se mostrou determinante, pois limiares mais baixos tendem a reagir mais cedo e conter o crescimento do pico, ao passo que limiares mais altos permitem maior acúmulo antes da resposta; a quantidade de réplicas adicionadas por acionamento aparece como ajuste complementar, com ganhos mais claros em cargas moderadas e limitações sob ataques mais intensos. Na prática, o modelo pode ser usado como uma ferramenta de apoio à configuração antes da implantação ou de mudanças operacionais, permitindo comparar cenários "what-if" de forma controlada: testar diferentes limiares e tempos de provisão para selecionar combinações que reduzam o pico de ocupação e, principalmente, minimizem o tempo acima de um nível máximo aceitável definido em SLA. Além disso, ele ajuda a identificar regimes em que adicionar mais réplicas deixa de trazer retorno, sinalizando a necessidade de aumentar capacidade base, ajustar o mecanismo de detecção/acionamento ou adotar políticas alternativas. Como trabalhos futuros pretendemos analisar outros fatores como disponibilidade de recursos, tolerância a falhas e sustentabilidade.

Referências

Bremler-Barr, A., Brosh, E., and Sides, M. (2017). Ddos attack on cloud auto-scaling mechanisms. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE.

Bremler-Barr, A., Czeizler, M., Levy, H., and Tavori, J. (2024). Exploiting miscoordination of microservices in tandem for effective ddos attacks. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 231–240. IEEE.

Corrêa, J. H. G., Junior, E. A. S., Fonseca, I. E., Nigam, V., Ribeiro, M. R., and Villaça, R. S. (2019). Selectivity and autoscaling as complementary defenses for ddos protection to cloud services. In *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pages 1–3. IEEE.

David, R. B. (2021). Kubernetes auto-scaling: Yoyo attack vulnerability and mitigation. Master's thesis, Reichman University (Israel).

Eriksson, L., Johansson, E., Kettaneh-Wold, N., Wikström, C., and Wold, S. (2000). Design of experiments. *Principles and Applications, Learn ways AB, Stockholm*.

g1 (2025). Caixa atrasa sorteio da mega da virada e apostadores reclamam nas redes. Acesso em 3 jan. 2026.

Huang, K., Peng, G., and Mehmood, F. (2025). A mathematical model of a novel proportional control scheme for mitigating ddos attacks in tcp/aqm-npc networks. *Computer Networks*, page 111893.

Khan, I. U., Abdollahi, A., Alturki, R., Alshehri, M. D., Ikram, M. A., Alyamani, H. J., and Khan, S. (2021). Intelligent detection system enabled attack probability using markov chain in aerial networks. *Wireless Communications and Mobile Computing*, 2021(1):1542657.

Kumari, P. and Jain, A. K. (2024). Timely detection of ddos attacks in iot with dimensionality reduction. *Cluster Computing*, 27(6):7869–7887.

Maciel, P. R. M. (2023). *Performance, reliability, and availability evaluation of computational systems, volume I: performance and background*. Chapman and Hall/CRC.

Mura, I. (2015). Detailed state probability distribution of infinite servers queues with phase-type distributed service times. *Revista Ontare*, 3(1):29–54.

Mutar, M. H., El Fawal, A. H., Nasser, A., and Mansour, A. (2024). Predicting the impact of distributed denial of service (ddos) attacks in long-term evolution for machine (lte-m) networks using a continuous-time markov chain (ctmc) model. *Electronics*, 13(21):4145.

Poojara, S. R., Dehury, C. K., Jakovits, P., and Srirama, S. N. (2022). Serverless data pipeline approaches for iot data in fog and cloud computing. *Future Generation Computer Systems*, 130:91–105.

Santos, L., Nguyen, T. A., and Silva, F. A. (2024). Internet of medical things: a performability performance analysis. *International Journal of Computer Applications in Technology*, 75(1):35–47.

Silva, B., Matos, R., Callou, G., Figueiredo, J., Oliveira, D., Ferreira, J., Dantas, J., Lobo, A., Alves, V., and Maciel, P. (2015). Mercury: An integrated environment for performance and dependability evaluation of general systems. In *Proceedings of industrial track at 45th dependable systems and networks conference, DSN*, pages 1–4.

Wang, D., Chen, X., Wang, Q., Wang, S., Xu, F., and Zheng, T. (2022). Autoscaling cracker: an efficient asymmetric ddos attack on serverless functions. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 4179–4184. IEEE.

Wei, W., Song, H., Wang, H., and Fan, X. (2017). Research and simulation of queue management algorithms in ad hoc networks under ddos attack. *Ieee Access*, 5:27810–27817.

Xie, L., Yuan, B., Yang, H., Hu, Z., Jiang, L., Zhang, L., and Cheng, X. (2024). Mrfm: A timely detection method for ddos attacks in iot with multidimensional reconstruction and function mapping. *Computer standards & interfaces*, 89:103829.