Arquitetura Distribuída para Monitoramento de Incêndios: Uma Abordagem com Redes de Petri Estocásticas

Arthur Sabino[1], Luiz Nelson Lima[1], Vandirleya Barbosa[1], Leonel Feitosa[1], Leonardo Freitas[2] Marcos F. Caetano [2], Priscila Solis Barreto[2], Francisco Airton Silva[1]

[1]Laboratório de Pesquisa Aplicada a Sistemas Distribuídos (PASID), Universidade Federal do Piauí (UFPI)

[2]Departamento de Ciência da Computação, Universidade de Brasília (UnB)

{arthursabino, luizznelson,vandirleya.barbosa}@ufpi.edu.br

{leonelfeitosa,faps}@ufpi.edu.br

{mfcaetano, pris}@unb.br, contato.leonardobbf@gmail.com

Resumo. *Os incêndios florestais representam uma séria ameaça ambiental, climática e social, exigindo sistemas de monitoramento com alta capacidade de resposta e resiliência. Este trabalho apresenta a modelagem e avaliação de desempenho de um sistema real de monitoramento de incêndios, baseado em Redes de Petri Estocásticas (SPN) e em uma arquitetura distribuída orientada a microsserviços escaláveis. O modelo SPN captura o comportamento dinâmico de componentes como Frame Producer, Frame Consumer, Rules Manager, Event Manager e Mosquitto, permitindo a análise detalhada de métricas críticas como tempo médio de resposta, throughput, utilização de recursos e probabilidade de descarte. A abordagem proposta evidencia como diferentes configurações impactam o desempenho, demonstrando, por exemplo, que a adoção de 4 instâncias no Frame Consumer reduz significativamente o MRT, além de evitar sobrecarga e permitir uma reserva de recursos. Esses resultados fornecem subsídios técnicos para a otimização e o dimensionamento de infraestruturas de monitoramento em tempo real.*

Abstract. *Forest fires represent a serious environmental, climatic and social threat, requiring monitoring systems with high response capacity and resilience. This paper presents the modeling and performance evaluation of a real fire monitoring system, based on Stochastic Petri Nets (SPN) and a distributed architecture oriented towards scalable microservices. The SPN model captures the dynamic behavior of components such as Frame Producer, Frame Consumer, Rules Manager, Event Manager and Mosquitto, allowing detailed analysis of critical metrics such as average response time, throughput, resource utilization and discard probability. The proposed approach highlights how different configurations impact performance, demonstrating, for instance, that adopting 4 instances in the Frame Consumer significantly reduces the MRT, while also preventing overload and allowing resource reservation. These results provide technical support for the optimization and sizing of real-time monitoring infrastructures.*

1. Introdução

Incêndios florestais estão entre os desastres naturais mais destrutivos, ameaçando a biodiversidade, comprometendo ecossistemas e impactando diretamente comunidades humanas. De março de 2023 a fevereiro de 2024, mais de 3,9 milhões de km² foram queimados globalmente [Jones et al. 2024]. Esses eventos, cuja origem pode ser natural ou antrópica [Hoover and Hanson 2023], intensificam as emissões de gases de efeito estufa e agravam os efeitos das mudanças climáticas. Com a intensificação e recorrência desses eventos, a demanda por sistemas inteligentes de monitoramento e resposta em tempo real tornou-se crítica [Mohammed et al. 2024].

Os sistemas de monitoramento de incêndios enfrentam desafios significativos para garantir detecção eficiente e resposta ágil, especialmente em vastas áreas florestais e regiões urbanas densas. A propagação rápida e imprevisível do fogo exige alocação otimizada de recursos, pois falhas na distribuição podem comprometer a identificação precoce dos incêndios, resultando em atrasos críticos na contenção e aumento dos danos ambientais, econômicos e humanos. Assim, aprimorar continuamente esses sistemas é essencial para mitigar impactos e garantir uma resposta eficaz às emergências.

A abordagem baseada em SPN permite avaliar o comportamento do sistema sob diferentes condições operacionais, contribuindo para o aprimoramento das infraestruturas de monitoramento. [Chen and Ha 2018]. Este trabalho propõe uma modelagem formal baseada em SPN de um sistema de monitoramento de incêndios. A arquitetura, composta por microsserviços distribuídos, é modelada com precisão para capturar os atrasos, gargalos e limitações de capacidade de cada componente. Com isso, torna-se possível analisar quantitativamente como diferentes configurações — como o número de instâncias de consumidores — impactam o desempenho geral do sistema. A análise foca em métricas como tempo médio de resposta (MRT), throughput (TP), probabilidade de descarte (DP) e utilização de recursos, com especial atenção ao MRT, essencial para aplicações sensíveis ao tempo.

Este artigo está organizado da seguinte maneira. A Seção 2 apresenta uma tabela comparativa com os trabalhos relacionados. A Seção 3 descreve a arquitetura utilizada. A Seção 4 detalha o modelo SPN proposto e seus componentes. Os resultados das simulações são discutidos na Seção 5. Por fim, a Seção 6 traz as conclusões do estudo.

2. Trabalhos Relacionados

Esta seção apresenta os trabalhos relacionados à avaliação de desempenho de sistemas de monitoramento de incêndios. Os métodos de avaliação incluem experimentos, simulações e modelagem, como a modelagem SPN utilizada por [Sabino et al. 2024] no contexto florestal. Trabalhos como [Abdusalomov et al. 2022], voltado para notificação de incêndios para auxiliar pessoas com deficiência visual, e [Talaat and Zain 2023], que aborda a detecção de incêndios em cidades inteligentes, utilizam experimentos para validar suas propostas. Já [Reddy et al. 2024] e [Al-Dhief et al. 2022] recorrem a simulações para avaliar protocolos e algoritmos em cenários florestais e urbanos. Esses métodos permitem analisar diversas métricas, como precisão, consumo de energia e taxa de entrega de pacotes.

As métricas avaliadas variam conforme o foco dos estudos. No contexto florestal, [Mukhia et al. 2023] avalia indicadores como força do sinal e taxa de entrega de pacotes em redes de sensores LoRaWAN. Em contraste, em ambientes urbanos, [Zhai 2024] e [Papaioannou et al. 2021] analisam a eficiência no processamento de dados e o consumo de energia, respectivamente. Alguns estudos destacam a utilização de microsserviços, como [Talaat and Zain 2023] e [Mukhia et al. 2023], enquanto a maioria não adota essa arquitetura. No que diz respeito à validação, trabalhos como [Zhang et al. 2023], no monitoramento agrícola, e [Shri et al. 2024, Dinesh et al. 2024], no contexto florestal, utilizam dados reais para corroborar suas análises.

Nosso trabalho se destaca ao combinar modelagem SPN de um sistema de monitoramento de incêndios, além da adoção de uma arquitetura baseada em microsserviços para o monitoramento florestal, o que oferece vantagens como escalabilidade e tolerância a falhas. Avaliamos métricas como MRT, TP, DP e utilização de recursos, permitindo a análise do comportamento do sistema sob diferentes cargas de trabalho. A escolha por SPN se justifica por sua capacidade de representar simultaneamente aspectos temporais, concorrência, bloqueios e limitação de capacidade, características comuns em sistemas distribuídos com filas assíncronas. Diferente de abordagens baseadas exclusivamente em simulação discreta ou teoria de filas, o modelo SPN permite capturar dinamicamente a interação entre múltiplos componentes e avaliar métricas com granularidade. Além disso, o modelo é expansível para incluir falhas, interrupções ou políticas de prioridade, o que o torna adequado para cenários realistas e críticos como o monitoramento de incêndios.

3. Visão Geral da Arquitetura

A arquitetura do sistema distribuído modelado neste trabalho é composta por microsserviços desacoplados, responsáveis por diferentes etapas do processamento de dados em tempo real. A Figura 1 ilustra os componentes principais e o fluxo de dados entre eles, desde a captura inicial até o armazenamento persistente. Nesta arquitetura, dois termos são empregados para representar as unidades de dados manipuladas pelos microsserviços: pacotes e quadros. O termo ”pacote”refere-se à estrutura de dados utilizada internamente para o transporte e processamento de informações.

No contexto da aplicação, um pacote pode conter diferentes tipos de conteúdo, como mensagens provenientes de sensores, notificações de eventos ou dados visuais. Especificamente no caso de dados oriundos de câmeras, utiliza-se o termo “quadro”para designar cada imagem individual capturada por esses dispositivos. Esses quadros são encapsulados em pacotes para serem transmitidos e processados pelo sistema. Assim, embora o modelo adote a terminologia de “pacote”como representação geral do fluxo de dados, o termo “quadro”destaca o aspecto visual desse conteúdo, sendo semanticamente equivalente a uma imagem no contexto de aplicações de monitoramento e detecção de eventos, como incêndios.

O fluxo de dados no sistema inicia no Frame Producer, responsável por gerar periodicamente pacotes de dados contendo informações relevantes, como imagens, leituras de temperatura ou sinais de fumaça. Os pacotes são enviados ao MinIO, que funciona como um armazenamento temporário e distribuído, oferecendo acesso seguro e eficiente aos dados para os demais microsserviços. A comunicação entre os componentes é orquestrada pelo Mosquitto, um broker MQTT que permite a troca assíncrona de mensagens via

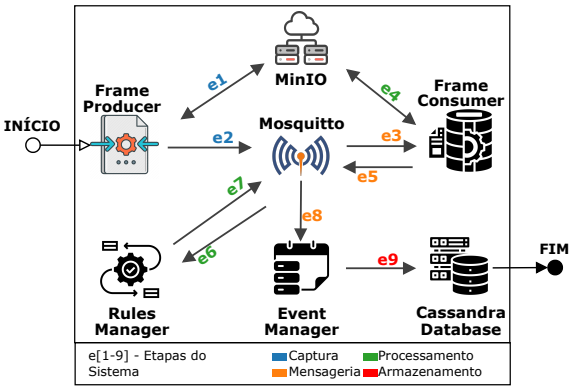


Figura 1. Arquitetura do sistema de monitoramento de incêndios florestais.

publicação e subscrição, promovendo o desacoplamento entre produtores e consumidores.

Em seguida, o Frame Consumer consome os pacotes armazenados no MinIO, executa um pré-processamento e publica os resultados no Mosquitto. O Rules Manager então aplica regras específicas de detecção — como limiares de temperatura ou padrões visuais — e classifica ou redireciona as mensagens com base em critérios definidos. Por fim, o Event Manager escuta os eventos publicados no Mosquitto e identifica quais devem ser persistidos. Esse componente interage com o banco de dados Cassandra, onde os dados processados são armazenados de forma permanente para consultas futuras, auditoria ou geração de alertas automatizados.

4. Modelo SPN Proposto

Esta seção descreve um modelo baseado na arquitetura descrita na Seção 3, detalhando o comportamento dos componentes do sistema. A Figura 2 ilustra a estrutura do modelo que integra a produção e o consumo de pacotes com a gestão de regras e eventos. O fluxo inicia com o Frame Producer, representado pelo lugar `Pafp`, onde pacotes são gerados a partir dos quadros capturados pelas câmeras. A geração é modelada pela transição temporizada `Tafp`, que introduz um atraso. Os pacotes são então armazenados em uma fila intermediária, representada por `Paqfp`.

A transição imediata `Ti1` move os pacotes da fila para o lugar `PfpMI`, que simboliza a preparação para o armazenamento no MinIO. O processo de armazenamento é representado pela transição temporizada `T1`, que leva os pacotes para o lugar `Pfp`. Em seguida, os pacotes são encaminhados para o broker Mosquitto, representado pela transição `T2` e o lugar `Pmq1`, que modela o enfileiramento inicial dos pacotes dentro do broker. Dentro do Mosquitto, o fluxo passa por três etapas de enfileiramento e processamento, modeladas pelos lugares `Pcmq1`, `Pcmq2` e `Pcmq3`. As transições `T3` e `T6` controlam a movimentação entre essas filas, simulando a lógica de entrega de mensagens assíncronas e a limitação da capacidade do broker.

O Frame Consumer consome os pacotes armazenados no MinIO, começando pelo lugar `PfcMI`, onde os pacotes são preparados para o consumo. A transição `T4` modela o tempo necessário para o consumo, movendo os pacotes para `PfcPM`. A capacidade máxima do Frame Consumer é indicada pela marcação `CFC` no lugar `Pcfc`, que controla

quantos pacotes podem ser processados simultaneamente. Após o consumo, os pacotes são processados pelo Rules Manager, que aplica regras específicas a cada pacote. O lugar `PrmAP` representa os pacotes prontos para o processamento de regras, enquanto a transição temporizada `T8` modela o tempo de aplicação dessas regras. Os pacotes então avançam para o lugar `PrmF`, que indica a conclusão do processamento. Finalmente, os pacotes chegam ao Event Manager, responsável por armazenar os dados no banco de dados Cassandra. Este processo é modelado pelo lugar `Pem`, com o tempo de armazenamento sendo representado pela transição temporizada `T11`. A capacidade do Event Manager é gerida pelo lugar `Pcem`, que assegura que o sistema funcione dentro de seus limites de capacidade.

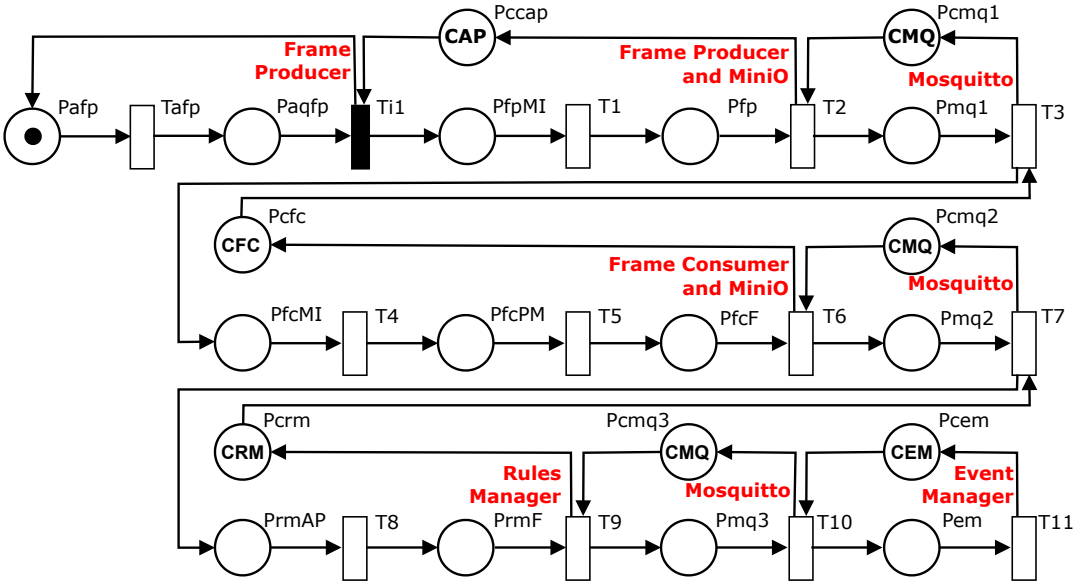


Figura 2. Modelo SPN proposto.

4.1. Métricas

O modelo permite calcular as seguintes métricas: tempo médio de resposta, a taxa de utilização do Mosquitto e do Frame Consumer, a probabilidade de descarte de pacotes e a vazão do sistema. A seguir, detalhamos o processo para a obtenção de cada uma dessas métricas usando o modelo SPN.

A Equação 1 mostra como é calculada a vazão do sistema do modelo. A Vazão (TP) é uma forma quantitativa que indica o número de solicitações processadas com sucesso pelo sistema em um determinado tempo. A métrica de TP é importante para avaliar a capacidade de processamento e a eficiência do sistema no processamento de dados. No modelo, a vazão é calculada pela esperança de se terem tokens na fila que representa os pacotes prontos para armazenamento no Cassandra, dividido pelo tempo de registro para esses tokens saírem desse lugar, ou seja, o tempo (t) da transição T11.

$$\mathbf{TP} = \left(\frac{E\{\#Pem\}}{t(T11)} \right) \tag{1}$$

A Equação 2 define o cálculo do tempo médio de resposta (MRT). O MRT pode ser derivado da lei de Little [Maciel 2023]. A Lei de Little, uma fórmula amplamente

utilizada em teoria de filas e sistemas de operações, estabelece uma relação fundamental entre três parâmetros principais: o número médio de itens em um sistema (L), a taxa média de chegada de itens ao sistema (λ) e o tempo médio que um item permanece no sistema (W), no nosso caso chamamos W de MRT. Formalmente, a lei é expressa como L=λ W. Essa relação é válida para sistemas em estado estacionário, independentemente da distribuição específica de chegada ou serviço, desde que as taxas sejam estáveis. O MRT, portanto, quantifica o tempo médio necessário para que um pacote de frames seja processado e registrado no Cassandra. O cálculo do MRT envolve a soma das esperanças de existência de tokens em todos os lugares que representam filas ou processos, dividido pela vazão na extremidade final do sistema (Cassandra).

$$\textbf{MRT} = \left(\frac{E\{\#PfpMI\} + E\{\#Pfp\} + E\{\#Pmq1\} + \cdots + E\{\#Pem\}}{TP} \right) \quad (2)$$

As Equações 3 e 4 explicam os cálculos do percentual de utilização (UN). A utilização mede a proporção do tempo em que os recursos disponíveis estão efetivamente em uso. No modelo proposto, é possível determinar a utilização em diferentes partes do sistema, permitindo configurar adequadamente os equipamentos para os cenários estudados. Neste trabalho, focamos na utilização do Mosquitto e do Frame Consumer. A fórmula considera a expectativa do número de tokens presentes na fila de processamento na borda, dividida pela capacidade de processamento do sistema de borda. O resultado é então multiplicado por 100 para expressar o valor em porcentagem.

$$\textbf{CMQ Utilization} = \left(\frac{E\{\#Pmq1\} + E\{\#Pmq2\} + E\{\#Pmq3\}}{CMQ} \right) \times 100 \quad (3)$$

$$\textbf{FC Utilization} = \left(\frac{E\{\#PfcMI\} + E\{\#PfcPM\} + E\{\#PfcF\}}{CFC} \right) \times 100 \quad (4)$$

A Equação 5 define como calcular a probabilidade de descarte (DP). Esta probabilidade mede a chance de solicitações serem descartadas pelo sistema devido à sobrecarga, ou incapacidade de processamento. A métrica DP é crucial para avaliar a capacidade do sistema de processar grandes volumes de dados, prevenindo a perda de informações importantes para o combate aos incêndios. No sistema proposto, analisamos a chance de perda de pacotes no enfileiramento do Frame Producer para o MinIO e na capacidade genérica constante configurada para a simulação. A probabilidade de descarte é calculada com base na probabilidade *P* de não haver tokens em um determinado lugar. O estudo verifica a probabilidade de ausência de tokens no enfileiramento do MinIO e na capacidade genérica, pois, se isso acontecer, todo o sistema subsequente estará sobrecarregado. O valor final é multiplicado por 100 para expressar a probabilidade em porcentagem.

$$\textbf{DP} = (P\{(\#Pccap = 0) \text{ AND } (\#Paqfp > 0)\}) \times 100 \quad (5)$$

5. Estudo de Caso

Esta seção apresenta os resultados das simulações realizadas para avaliar o desempenho da arquitetura modelada do sistema de monitoramento de incêndios. O foco está em como diferentes configurações, especialmente o número de instâncias do Frame Consumer, influenciam métricas críticas como tempo de resposta, utilização de recursos, probabilidade de descarte e throughput. As simulações estacionárias foram conduzidas com margem de erro de aproximadamente 2%, utilizando a ferramenta Mercury (v5.0.1) [Silva et al. 2015].

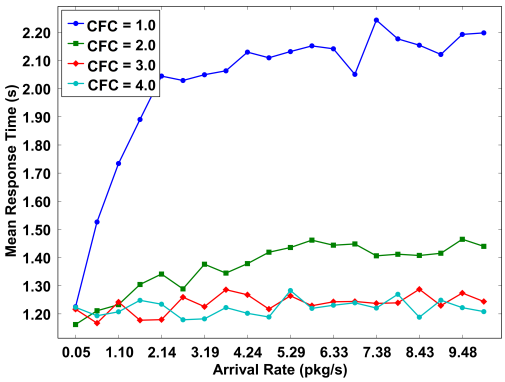
Apresentamos os resultados da variação da quantidade de instâncias do Frame Consumer para o tempo médio de resposta do sistema, a utilização de recursos do Frame Consumer, a utilização de recursos do Mosquitto, a probabilidade de descarte, e o throughput. A Figura 3a mostra que, à medida que a taxa de chegada aumenta, o tempo médio de resposta (MRT) cresce, sobretudo quando há apenas uma instância de Frame Consumer ($CFC = 1.0$) — alcançando até $2.20\ ms$ para taxas acima de $7.38\ pcts/s$. Para $CFC = [2.0,\ 3.0,\ 4.0]$, o MRT se mantém estável em taxas moderadas, mas aumenta após $5.29\ pcts/s$. Entre todas as configurações, $CFC = 4.0$ apresenta o melhor desempenho, mantendo o MRT abaixo de $1.40\ ms$ mesmo sob alta carga.

A Figura 3b detalha a utilização de recursos do Frame Consumer em diferentes taxas de chegada. A utilização de recursos cresce consideravelmente com o aumento da taxa de chegada. Com $CFC = 1.0$, a utilização atinge cerca de 90% em taxas de chegada a partir de $7.38\ pcts/s$, sugerindo que o sistema está sobrecarregado. Já para $CFC = [2.0,\ 3.0\ ,\ 4.0]$, a utilização de recursos é mais equilibrada, com $CFC = 4.0$ mantendo a utilização em torno de 35% a 45% na maior parte das condições, o que indica um melhor balanceamento da carga com mais instâncias do Frame Consumer.

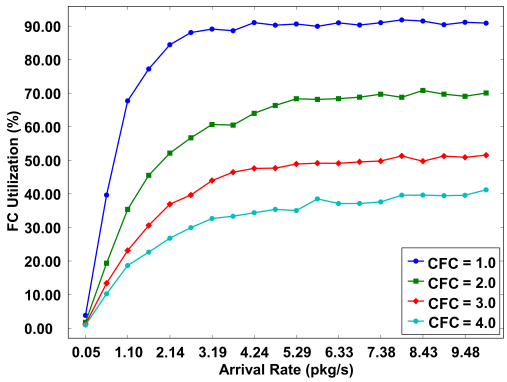
A Figura 3c mostra que a utilização do Mosquitto também cresce com a carga, atingindo cerca de 75% para $CFC = 1.0$ com taxas acima de $5.29\ pcts/s$. Em contraste, com $CFC = 4.0$, a utilização permanece abaixo de 40%, evidenciando que o aumento do número de consumidores reduz a pressão sobre o middleware de mensageria.

A Figura 3d mostra a probabilidade de descarte à medida que a taxa de chegada aumenta. Com $CFC = 1.0$, a probabilidade de descarte aumenta rapidamente, alcançando até 80% em taxas de chegada próximas de $9.48 pcts/s$. Quando mais instâncias são adicionadas ($CFC = 2.0,\ 3.0\ e\ 4.0$), a probabilidade de descarte diminui significativamente. Com $CFC = 4.0$, a probabilidade de descarte permanece quase nula em taxas de chegada mais baixas, como 3.19 pcts/s, demonstrando a eficácia de aumentar o número de instâncias para evitar perdas.

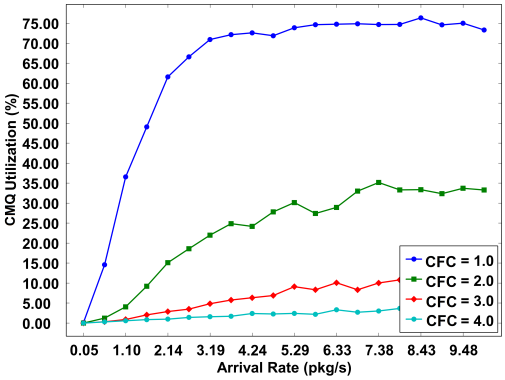
Por fim, a Figura 3e apresenta o throughput em função da taxa de chegada para diferentes configurações de CFC. O throughput aumenta de forma consistente até cerca de $6.33\ pcts/s$, onde começa a se estabilizar, especialmente em configurações com mais instâncias. A configuração com $CFC = 4.0$ atinge o maior throughput, chegando a aproximadamente $2.10\ pcts/s$ em taxas de chegada entre 7.38 e $9.48\ pcts/s$, o que indica uma maior capacidade de processamento e melhor distribuição de carga em comparação com configurações com menos instâncias.



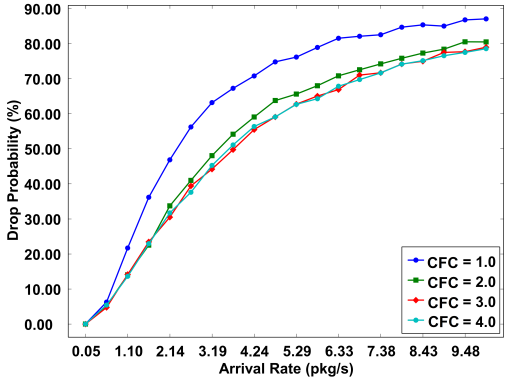
(a) Tempo Médio de Resposta do Sistema.



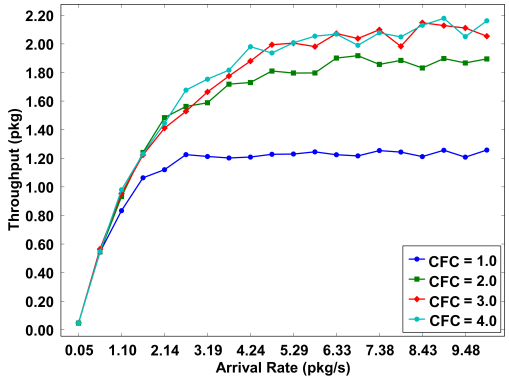
(b) Utilização de Recursos do Frame Consumer.



(c) Utilização de Recursos do Mosquitto.



(d) Probabilidade de Descarte do Sistema.



(e) Vazão do Sistema.

Figura 3. Resultados do estudo da variação do número de consumidores.

Os resultados indicam que a configuração com $CFC = 4.0$ proporciona um equilíbrio eficaz entre desempenho e utilização de recursos. Com essa configuração, o sistema atinge baixa latência (MRT $\approx 1.20\ ms$), alta taxa de processamento (TP $\approx 2.10\ pcts/s$) e mínima probabilidade de descarte, mesmo sob cargas elevadas. Embora o aumento no número de consumidores além desse ponto possa parecer vantajoso, os ganhos marginais tornam-se insignificantes diante do custo adicional de infraestrutura.

Assim, $CFC = 4.0$ pode ser considerado um ponto ótimo de capacidade para o cenário analisado.

O ponto ótimo encontrado nesse estudo de caso não deve ser interpretado como um limite fixo, mas sim como uma referência para o dimensionamento inicial. Em cenários com aumento sazonal de tráfego, alterações na topologia do sistema, falhas de componentes ou requisitos de alta disponibilidade, a adição de mais instâncias pode ser estratégica. A modelagem proposta permite avaliar com precisão tais decisões, funcionando como uma ferramenta para planejamento dinâmico de capacidade, adaptável a diferentes demandas operacionais e orçamentárias.

Embora o modelo SPN seja eficaz para sistemas homogêneos, sua aplicação em sistemas distribuídos enfrenta desafios devido a falhas de componentes, latências variáveis, comportamento não determinístico e condições climáticas adversas. Para superar essas limitações, propõe-se a expansão do modelo para incluir tais cenários.

6. Conclusão

Este artigo propôs um modelo baseado em SPN para avaliar o desempenho de um sistema real de monitoramento de incêndios florestais. A modelagem permitiu analisar métricas críticas, como MRT, TP, DP e utilização de recursos, sob diferentes configurações de capacidade. Os resultados demonstraram que o número de instâncias do Frame Consumer e o tempo de atraso na geração de pacotes pelo Frame Producer são os fatores mais impactantes na performance geral do sistema. Observou-se também que o balanceamento dos recursos entre componentes — especialmente no Mosquitto e no Rules Manager — é essencial para garantir eficiência e evitar gargalos. A modelagem SPN mostrou-se eficaz para representar o comportamento do sistema, oferecendo suporte à tomada de decisão no planejamento e dimensionamento da infraestrutura. O modelo proposto assume homogeneidade entre os componentes e ausência de falhas ou degradação de desempenho. Em ambientes reais, sensores e serviços podem apresentar latências variáveis, falhas temporárias e comportamento não determinístico, o que pode afetar diretamente o desempenho do sistema. Além disso, a validação do modelo com dados reais ainda é uma etapa futura, prevista para reforçar a aderência do modelo às condições operacionais reais. Como próximos passos, pretende-se expandir a modelagem SPN para representar cenários com falhas, heterogeneidade e mecanismos de tolerância a falhas, permitindo uma avaliação mais robusta e alinhada com os desafios enfrentados em ambientes de monitoramento distribuído em campo.

Referências

Abdusalomov, A. B., Mukhiddinov, M., Kutlimuratov, A., and Whangbo, T. K. (2022). Improved real-time fire warning system based on advanced technologies for visually impaired people. *Sensors*, 22(19):7305.

Al-Dhief, F. T., Muniyandi, R. C., Sabri, N., Hamdan, M., Latiff, N. M. A., Albadr, M. A. A., Khairi, M. H. H., Hamzah, M., and Khan, S. (2022). Forest fire detection using new routing protocol. *Sensors*, 22(20):7745.

Chen, L. and Ha, W. (2018). Reliability prediction and qos selection for web service composition. *International Journal of Computational Science and Engineering*, 16(2):202–211.

Dinesh, M., Dalei, J., and Ray, K. C. (2024). An optimized lightweight convolutional neural network framework and hardware system for forest fire monitoring. *Available at SSRN 4913945*.

Hoover, K. and Hanson, L. A. (2023). *Wildfire statistics*. Congressional Research Service.

Jones, M. W., Kelley, D. I., Burton, C. A., Di Giuseppe, F., Barbosa, M. L. F., Brambleby, E., Hartley, A. J., Lombardi, A., Mataveli, G., McNorton, J. R., et al. (2024). State of wildfires 2023–24. *Earth System Science Data Discussions*, 2024:1–124.

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

Mohammed, M. S., Abbas, A. H., and Abdullah, N. A. (2024). Intelligent surveillance system for fire detection using yolov8. *Iraqi Journal for Computers and Informatics*, 50(1):105–122.

Mukhia, R., Sarambage Jayarathna, K. G., and Lertsinsrubtavee, A. (2023). Performance evaluation of lorawan forest fire monitoring network in the wild. In *Proceedings of the 18th Asian Internet Engineering Conference*, pages 96–104.

Papaioannou, A., Verikios, P., Kouzinopoulos, C. S., Ioannidis, D., and Tzovaras, D. (2021). A low-power embedded system for fire monitoring and detection using a multilayer perceptron. In *2021 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE.

Reddy, P. D. K., Margala, M., Shankar, S. S., and Chakrabarti, P. (2024). Early fire danger monitoring system in smart cities using optimization-based deep learning techniques with artificial intelligence. *Journal of Reliable Intelligent Environments*, 10(2):197–210.

Sabino, A., Lima, L. N., Brito, C., Feitosa, L., Caetano, M. F., Barreto, P. S., and Silva, F. A. (2024). Forest fire monitoring system supported by unmanned aerial vehicles and edge computing: a performance evaluation using petri nets. *Cluster Computing*, pages 1–21.

Shri, A. L., Swathiha, R., Ismail, M. M., Krithiga, M., Meenakshi, B., and Kathir, M. (2024). Iot-enhanced forest fire monitoring and notification system. In *2024 5th International Conference on Image Processing and Capsule Networks (ICIPCN)*, pages 860–865. IEEE.

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.

Talaat, F. M. and Zain, H. (2023). An improved fire detection approach based on yolo-v8 for smart cities. *Neural Computing and Applications*, 35(28):20939–20954.

Zhai, Y. (2024). Design and optimization of smart fire iot cloud platform based on big data technology. In *2024 International Conference on Electrical Drives, Power Electronics & Engineering (EDPEE)*, pages 843–846. IEEE.

Zhang, W., Zhao, B., Gao, S., Zheng, Y., Zhou, L., and Liu, S. (2023). Development of cotton picker fire monitoring system based on ga-bp algorithm. *Sensors*, 23(12):5553.