Comparação de Desempenho entre Soluções de Interoperabilidade

João Marcelo U. de Alencar[1], **Regis P. Magalhães**[1,2], **Davi R. de Vasconcelos**[1,2], **Samir B. Chaves**[2], **João Victor C. de Oliveira**[1,2], **Anderson T. Bessa**[2], **Emanuel M. Rodrigues** [1,2]

[1]Campus de Quixadá – Universidade Federal do Ceará (UFC)
Av. José de Freitas Queiroz, 5003 – Cedro – Quixadá – CE 63902-580

[2]Insight Lab – Universidade Federal do Ceará (UFC)
Campus do Pici, Bloco 952, Centro de Ciências, DC - Pici, Fortaleza - CE, 60440-900

joao.marcelo@ufc.br
{regis, daviromero}@insightlab.ufc.br
{samirchaves, joao.oliveira}@insightlab.ufc.br
{anderson.bessa, emanuel.rodrigues}@insightlab.ufc.br

Abstract. *The interoperability of digital services allows governments to increase the reach of public policies, as in the example of Estonia, which offers broad access to its citizens through the X-Road solution. However, the team behind X-Road did not consider the context of computing clouds and microservices. This work proposes an architecture using the API Gateway Kong solution with the same security as X-Road. We perform a performance comparison between the two solutions. The results show that the Kong-based architecture presents superior performance, providing subsidies to improve it and directions for X-Road optimization.*

Resumo. *A interoperabilidade de serviços digitais permite a governos aumentar o alcance de políticas públicas, como no exemplo da Estônia, que através da solução X-Road oferece amplo acesso aos seus cidadãos. Entretanto, o X-Road não foi concebido levando em consideração o contexto de nuvens computacionais e microsserviços. Este trabalho propõe uma arquitetura utilizando a solução de API Gateway Kong que apresenta uma segurança semelhante ao que o X-Road e compara o desempenho entre as duas soluções. Os resultados mostram que a arquitetura baseada em Kong apresenta desempenho superior no tempo de resposta das invocações de serviços, fornecendo subsídios para aprimorá-la e direções para otimização do X-Road.*

1. Introdução

De acordo com [Ministério do Planejamento 2012], a interoperabilidade pode ser entendida como uma característica que se refere à capacidade de diversos sistemas e organizações trabalharem em conjunto (interoperar) de modo a garantir que pessoas, organizações e sistemas computacionais interajam para trocar informações de maneira eficaz e eficiente.

Governos, em qualquer esfera, são formados por secretarias ou órgãos com autonomia, cada um com sua equipe e necessidades particulares de tecnologia da informação.

Desta forma, interoperabilidade é um conceito de gestão, mas para sua realização é necessário que os órgão participantes estabeleçam acordos sobre as tecnologias e padrões utilizados para a implantação de um ecossistema de sistemas computacionais interoperáveis.

À medida que as redes locais e a Internet amadureceram, vários *frameworks* ou *middlewares* surgiram como potenciais candidatos para habilitar a interoperabilidade. Soluções como CORBA e Java RMI [Raj 1998] obtiveram sucesso em ambientes ligados por redes locais ou de abrangência metropolitana. Com o advento da Internet e do protocolo HTTP, serviços *web* baseados na pilha SOAP/WSDL [Curbera et al. 2002] e no padrão REST [Pautasso 2014] se tornaram as opções mais adequadas para ambientes com redes locais independentes, protegidas por *firewalls* mas com o protocolo HTTP liberado. No contexto de aplicações *cloud native* [Gannon et al. 2017], a ubiquidade do protocolo HTTP e a simplicidade das arquiteturas REST possibilitaram a criação de aplicações escaláveis em nível global, com resiliência e atualizações sem disrupção de serviços.

Por volta do ano 2000, a Estônia buscou maneiras de agilizar sua economia e integração com a União Europeia. O caminho encontrado por essa nação do leste europeu foi a digitalização dos seus serviços públicos [Anthes 2015], iniciando um processo de interoperabilidade entre os sistemas governamentais. O ambiente *X-Road* é uma camada de comunicação segura na Internet através da qual participantes podem publicar serviços. Foi criado utilizando a pilha de *web services* SOAP/WSDL, a tecnologia mais proeminente na época. No decorrer dos anos 2000, este *middleware* ganhou suporte para publicação de APIs REST pelas entidades participantes. A maturidade do *X-Road* atraiu o interesse de pesquisadores no Brasil [Almeida and Costa 2021] e no mundo [NIIS, Nordic Institute for Interoperability Solution 2020] como alternativa para criação de soluções de governo digital.

Apesar do suporte à REST, o *X-Road* não foi desenvolvido considerando microsserviços e aplicações *cloud native*. Extensões que apresentam essas tecnologias foram adicionadas no decorrer de novas versões, entretanto a estrutura interna do *middleware* ainda é monolítica. No cenário atual das nuvens computacionais, é importante considerar o desenvolvimento de uma solução que apresente as mesmas funcionalidades e garantias de segurança do *X-Road*, mas construída utilizando um arcabouço de *software* contextualizado no universo de microsserviços.

Este trabalho analisa a possibilidade de a partir da solução do *gateway Kong* [1], construir uma solução capaz de apresentar a mesma segurança e confidencialidade na comunicação que o *X-Road* oferece. As contribuições deste trabalho são: (i) comparação de desempenho entre a solução de integração e interoperabilidade *X-Road* e uma arquitetura proposta robusta e escalável com funcionalidade similar, usando o *API Gateway Kong*; (ii) projeto de uma arquitetura de interoperabilidade em um cenário de serviços públicos no Brasil.

O restante desse trabalho é organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica e descrição da arquitetura de cada solução. Em seguida, na Seção 3, uma análise de desempenho na invocação de uma API REST avalia a sobrecarga de processamento em cada *middleware*. Por fim, a Seção 4 apresenta a conclusão das

[1] `https://konghq.com/kong/`

análises realizadas no contexto deste trabalho e lista trabalhos futuros.

2. *Middlewares* para Construção de Barramentos de Integração

Esta seção apresenta as arquiteturas gerais do *X-Road* e do *Kong*. Como o *X-Road* já é uma solução criada para interoperabilidade, sua arquitetura de referência já atende o propósitos deste trabalho. No caso do *Kong*, além do seu uso tradicional como *API Gateway*, este trabalho propõe uma arquitetura equivalente em segurança, no sigilo da comunicação, e interoperabilidade dos serviços em relação ao *X-Road*.

2.1. X-Road

O *X-Road* é um *software* de código aberto e solução de ecossistema que fornece troca de dados segura e unificada entre organizações [NIIS 2020]. A ideia básica do *X-Road* é que membros de um ecossistema trocam dados através de pontos de acesso (*Security Servers*) que implementam as mesmas especificações técnicas. Mesmo sendo código aberto, ingressar em um ecossistema *X-Road* envolve um processo de cadastro. Durante esse processo, a identidade de cada organização e ponto de acesso técnico são verificados usando certificados emitidos por uma Autoridade Certificadora (*Certification Authority* - CA) confiável. As identidades são mantidas centralizadas, mas todos os dados são trocados diretamente entre o produtor e o consumidor do serviço.

O roteamento de mensagens pelo *X-Road* é baseado nos identificadores de organizações e níveis de serviços que são mapeados para localizações físicas de rede dos serviços. Toda evidência sobre a troca de dados é armazenada localmente pelos atores da troca de dados e nenhum terceiro tem acesso aos dados. *Time-stamping* e assinaturas digitais garantem de forma conjunta o não repúdio dos dados enviados através do *X-Road*. Os registros fornecidos podem ser usados em procedimentos legais como evidência.

A Figura 1 apresenta uma visão simplificada de uma ecossistema formado por duas organizações, uma desempenhando o papel de Provedor de Serviços (*Service Provider*) e outra no papel de Consumidora de Serviços (*Service Consumer*). Um órgão no topo da hierarquia do governo estabelece um provedor *X-Road Central Services* que hospeda serviços gerais para cadastro e autenticação. Esse órgão também disponibiliza serviços de confiança (*Trust Services*), especificamente a Autoridade Certificadora e o serviço de marcação de tempo *time-stamping*, que podem estar implantados na mesma infraestrutura do *X-Road Central Services* ou serem serviços autônomos.

Todas mensagens enviadas pelo *X-Road* recebem marcação de tempo. O propósito da marcação é verificar a existência de itens de dados em determinado ponto do tempo. O *Security Server* usa o serviço de *time-stamping* para marcar todas as requisições de entrada/saída e respostas. O serviço de marcação deve implementar o protocolo suportado pelo *X-Road*, que suporta marcação em lote, o que reduz a carga do serviço de *time-stamping*.

Cada organização, ao fazer o cadastro no ecossistema, tem seus certificados emitidos e os configura em um servidor chamado *Security Server*, em execução na infraestrutura que tem controle administrativo. Dessa forma, todos os *Security Servers* do ecossistema podem trocar mensagens entre si com sigilo, autenticação e não repúdio. Esse canal seguro é representado pelo círculo no centro da Figura 1, sendo que a infraestrutura que o suporta é a Internet pública.

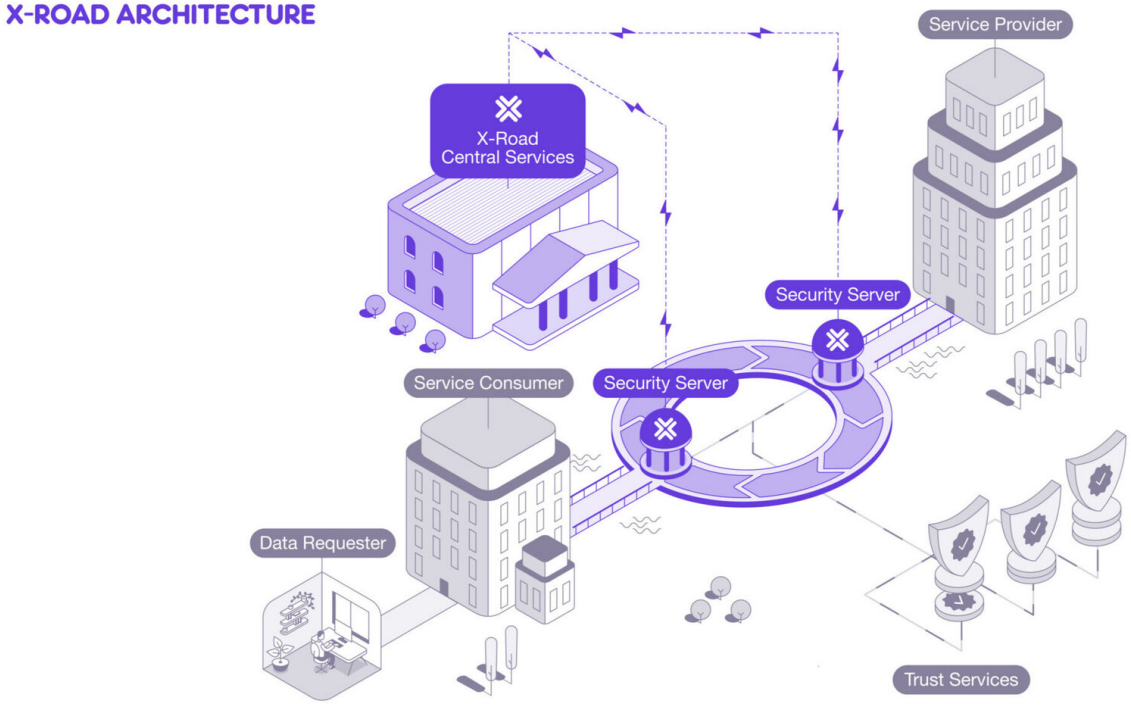


Figura 1. Arquitetura Geral do *X-Road* [NIIS 2020]

A invocação de um serviço através do *X-Road* inicia com o cadastro do mesmo pelo Provedor de Serviço (*Service Provider*) em seu *Security Server*. O serviço cadastrado é hospedado na infraestrutura do provedor, sendo que não precisa ser exposto à Internet pública, basta que seja acessível pelo servidor que executa o *Security Server*, pois este é que fará o roteamento das requisições. O Consumidor do Serviço, um subsistema que faz parte de uma organização já cadastrada no ecossistema, realiza a invocação do serviço cadastrado pelo Provedor não diretamente no *Security Server* do Provedor, mas sim no seu *Security Server* local (*Service Consumer*), que encaminha a mensagem para o Provedor. A resposta da requisição faz o caminho inverso, também usando o canal seguro.

O *X-Road* fornece funcionalidades de monitoramento e criação de relatórios que podem ser utilizadas para coletar relatórios de dados operacionais e informações de monitoramento técnicas do ecossistema. A informação pode ser utilizada para medir o uso de serviços individuais, entender dependências entre sistemas de informação e serviços diferentes, monitorar a saúde de serviços, monitorar as versões do *X-Road*, dentre outras informações. Cada membro pode acessar seus próprios dados, enquanto o mantenedor pode acessar todos os dados de qualquer membro.

2.2. Kong

O *Kong* implementa um padrão de arquitetura chamado *API Gateway*, apresentada na Figura 2. Este padrão busca agregar o acesso a diversos serviços em uma mesma porta de entrada. Isso traz várias vantagens, como a centralização da implementação de funções como autorização, autenticação, *logging*, *caching*, auditoria, entre outras, garantindo segurança e observabilidade no acesso ao conjunto inteiro de serviços. Com isso, muitas ideias de funções customizadas para domínios específicos surgem e o *Kong* possibilita a adição dessas extensões por meio de *plugins*.

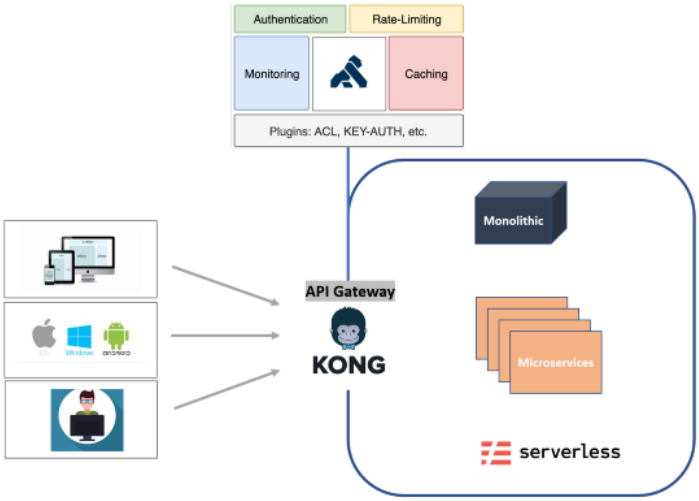


Figura 2. Arquitetura do Kong [Ramakani 2020]

Valendo-se das funcionalidades trazidas pelo *Kong* e do seu conceito de *plugin*, a arquitetura de barramento proposta aqui, descrita na Figura 3, surge como uma alternativa moderna às ferramentas de governança de APIs já existentes. Seguindo um paradigma de integração completamente distribuído e escalável, a nova arquitetura baseia-se em estratégias do estado da arte de aplicações distribuídas, como o *Service Mesh* [Richardson 2019].

Muito utilizado no contexto de microsserviços, o *Service Mesh* busca facilitar a implementação de vários padrões dessa arquitetura (como *Logging*, *Healh Check*, *Metrics*, entre outros) interceptando toda a comunicação entre os serviços internos de uma aplicação. Para isso, ele faz uso do mesmo conceito de *sidecar* aplicado aqui, no entanto, tratando-se de microsserviços, essa comunicação é geralmente feita dentro de uma mesma infraestrutura.

No ambiente de barramento de integração, cada serviço pode estar em uma rede diferente, o que traz um cenário mais amplo, onde a troca de mensagens é feita pela internet. Por isso, a comunicação entre os *sidecars* acontece de forma autenticada e protegida com a criptografia implementada no certificado SSL. Além disso, cada *sidecar* é acoplado a uma aplicação completa pertencente ao barramento, ao invés de estar acoplado a um único microsserviço no cenário tradicional.

Essa proposta visa ser aplicada em um cenário comum, porém complexo: diferentes aplicações que precisam enviar e receber dados de forma segura e escalável, de fácil gerenciamento e que permita o controle e observação do tráfego dessas informações. Para atender a esses requisitos, a ideia é que, dado uma API REST existente, uma *proxy sidecar* (implementado pelo *Kong*) seja acoplado a ela integrando-a ao *cluster* do barramento. Com isso, toda a comunicação entre as diferentes aplicações é interceptada pelos seus respectivos *sidecars* e todos eles comunicam-se com um Servidor Central, responsável por centralizar informações de todo o barramento.

Essa arquitetura garante uma comunicação segura e direta (ponta-a-ponta) ao passo que também permite uma visão global dos integrantes do barramento, dos seus serviços e da dinâmica de suas comunicações, podendo gerar *insights* sobre a demanda de

determinadas informações e auxiliando na tomada de decisões sobre o seu gerenciamento.

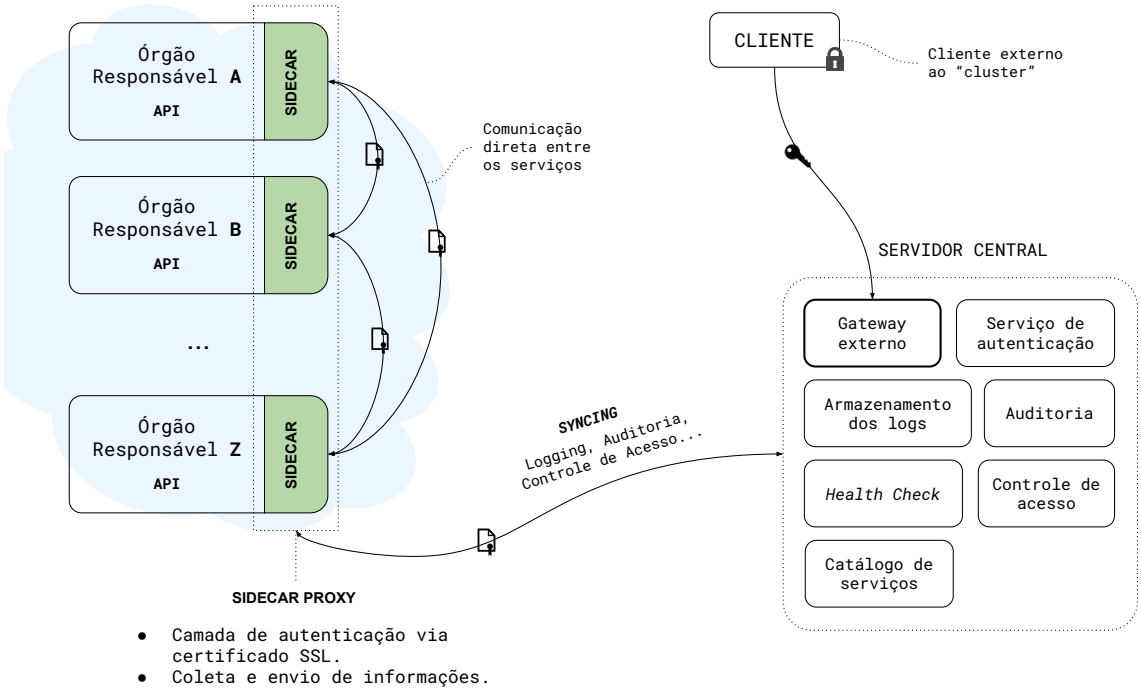


Figura 3. Arquitetura Distribuída Proposta com o *Kong* (Elaboração Própria)

2.2.1. Fluxo da arquitetura

A arquitetura proposta para os testes simula a troca de informações entre dois serviços, um que atua como cliente e o outro como servidor. O cliente possui o papel de fazer uma requisição para o *sidecar* (serviço do *Kong*) do servidor. Em comparação, com o *X-Road* pode-se dizer que o *sidecar* representa o *security server*.

Como o endereço do provedor está sujeito a mudanças, quando o cliente requisitar o servidor, ele irá consultar um outro serviço responsável por prover essa informação. Uma implementação de *software* responsável por isso chama-se *Consul*[2], um servidor DNS amplamente utilizado, que atua fornecendo a informação de acesso ao serviço, um endereço que esteja associado ao que o cliente deseja acessar. Quando a requisição é redirecionada ao serviço ocorre o processamento da requisição que irá depender da lógica interna da aplicação. Após isso a resposta é encaminhada ao cliente e o ciclo é concluído.

O Servidor Central, como é mostrado na Figura 3, constitui uma aplicação com diferentes componentes e, assim, pode desempenhar vários papéis. Por exemplo, o recurso de *Healch Check*, que mantém um status do funcionamento de cada serviço do barramento, é oferecido pelo *Consul*, em adição às funções já citadas. Além dele, existe o módulo de autenticação, responsável por autenticar e validar cada requisição feita entres os serviços, garantindo uma comunicação segura.

[2]https://www.consul.io/

Para fins de monitoramento, também existe um módulo responsável por receber e armazenar os *logs* das requisições, permitindo uma visão rica da carga de acessos em cada serviço, assim como provê recursos de auditoria. Visando atender a demandas de governança, também existe o módulo de Catálogo de Serviços, o qual permite ter uma visão geral de todos os serviços oferecidos pelos diferentes órgãos do barramento. Por fim, para trabalhos futuros, pensou-se na possibilidade de permitir um acesso externo ao barramento para clientes autenticados, o que será implementado pelo *gateway* externo.

2.3. Comparação do X-Road com a Arquitetura Proposta

A Tabela 1 estabelece uma comparação entre a arquitetura proposta neste trabalho para o uso do *Kong* e o *X-Road* em sua instalação nativa.

	X-Road	*Kong*
Serviços Internos	SOAP	REST
Serviços Disponibilizados	SOAP e REST	REST (nativo), SOAP e GraphQL(*plugins*)
Extensibilidade	Não	*Plugins*
Comunicação Sigilosa	Sim	Sim
Gestão de Identidades	Sim	Não
Interface de Gestão de API	Sim	Não
Execução na Nuvem	Máquinas Virtuais	Máquinas Virtuais e Contêineres
Licença	Código Aberto	Código Aberto, versão paga

Tabela 1. Características da arquitetura proposta neste trabalho com o Kong em relação ao *X-Road*.

O *X-Road* tem seu serviços internos implementados utilizando o protocolo SOAP, enquanto o *Kong* é estruturado em serviços REST. Ambas plataformas suportam registro de serviços SOAP e REST, porém por ter sua arquitetura interna também em REST, o *Kong* apresenta maior extensibilidade através da integração de *plugins*, permitindo acrescentar suporte a outros protocolos, como o *GraphQL*.

Na proposta atual, ambas soluções garantem a comunicação sigilosa. Entretanto, considerando a gestão de identidades, o *X-Road* é capaz de gerar certificados para membros e subsistemas, garantindo assim o não repúdio no ecossistema. A arquitetura baseada em *Kong* não tem essa funcionalidade. Em termos de facilidade de uso, o *X-Road* apresentam em cada *Security Server* uma interface gráfica para a gestão de APIs, enquanto o *Kong* oferece uma API para gestão, mas não tem interface.

No momento de decidir em qual plataforma computacional executar os serviços, o *Kong* possui imagens em contêineres testas e prontas para produção, mas também a possibilidade de executá-lo como serviço em uma instância de máquina virtual. O *X-Road* também oferece imagens, mas não são recomendadas para produção, apenas para testes e desenvolvimento. Ambas soluções são em código aberto, mas o *Kong* oferece uma versão paga com plano de suporte.

3. Avaliação de Desempenho na Invocação de APIs

O *X-Road* é uma solução de interoperabilidade baseada na implantação de *proxies* chamados de *Security Servers*, através dos quais membros devidamente registrados no *Central Server* podem realizar invocação segura de serviços. O *Kong* é um *API Gateway* otimizado para microserviços e arquiteturas distribuídas. Ambas soluções são consideradas para a definição de um Barramento de Interoperabilidade para serviços digitais.

Tanto o *X-Road* quanto o *Kong* atuam como uma camada de comunicação (*middleware*) entre um cliente que invoca um serviço de um provedor. Uma vez que adicionam funcionalidades como descoberta de serviços e comunicação segura, é natural que ocorra uma sobrecarga na comunicação, ou seja, uma invocação de serviços através dessas plataformas deve demorar mais do que uma invocação direta. Uma questão em aberto é qual das duas apresenta uma sobrecarga maior.

Um maior tempo gasto em comunicação acarreta recursos computacionais ociosos, gerando custo financeiro sem produção de valor, seja no cenário *on premises* ou em nuvem. Portanto, considera-se que analisar o impacto na sobrecarga de comunicação é um passo importante na escolha entre essas duas tecnologias.

3.1. Cenário de Experimentos

A Figura 4 representa o cenário de experimentos. São utilizadas duas máquinas físicas em um *cluster*, cada máquina física com duas máquinas virtuais. Os servidores físicos possuem processadores Intel(R) Core(TM) i7-10700F 2.90GHz e 64 GB de RAM, enquanto cada máquina virtual é configurada com 2 núcleos e 4 GB de RAM. O sistema operacional instalado tanto nas máquinas virtuais quanto físicas é o Ubuntu Linux 20.04 e para virtualização, o sistema *Proxmox Virtual Environment 7.0-13*[3] é utilizado. A comunicação entre qualquer ambiente virtual entre duas máquinas físicas acarreta troca de dados pela rede *Gigabit Ethernet*, mas entre duas máquinas virtuais em um mesmo hospedeiro físico, a comunicação é pela memória. O objetivo é isolar o processamento das mensagens de origem e destino ao mesmo tempo que minimiza o impacto da rede física, que não faz parte do objetivo deste trabalho avaliar.

No Provedor, há um serviço sem funcionalidade específica, apenas para realização dos testes. O serviço recebe um valor inteiro e responde uma quantidade de *kilobytes* equivalente ao valor informado. Na invocação direta, um processo em execução na máquina Cliente realiza a invocação no *endpoint* do serviço no Provedor, sendo que neste cenário há apenas a sobrecarga da comunicação em rede *Gigabit Ethernet*. Na invocação indireta, a comunicação ocorre através da camada de *middleware*.

A camada do *middleware* tem mais dois cenários. O primeiro considera o *middleware* como sendo dois *Security Servers* do *X-Road*. O segundo são duas instâncias do *Kong* fazendo o roteamento de mensagens. Pode-se, então, sumarizar os três cenários para a análise da seguinte forma:

1. Invocação Direta.
2. Invocação com o *X-Road* no papel de *Middleware*.
3. Invocação com o *Kong* no papel de *Middleware*.

[3] `https://www.proxmox.com/en/`

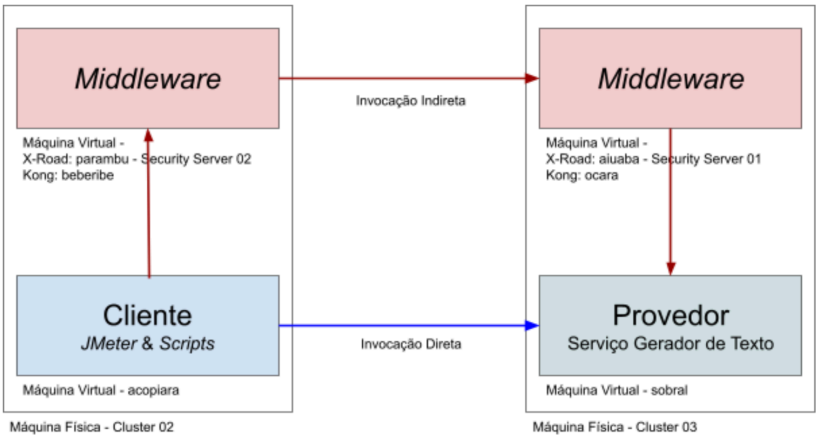


Figura 4. Cenário de Experimentos

3.2. Carga de Trabalho

Considerando o objetivo de realizar análise da sobrecarga de comunicação de uma implantação do *X-Road* ou do *Kong* em relação a invocação direta de um serviço, faz-se necessário definir uma carga de trabalho a ser submetida ao cenário implantado.

Este trabalho define duas classes de experimentos: serial e paralela. A versão serial realiza uma invocação por vez. A versão paralela faz quatro invocações ao mesmo tempo, sendo ajustado no JMeter [4] a divisão de oito HTTP *Requests* cada um com valor determinado de execução. É estabelecida uma mesma carga de trabalho a ser aplicada em cada um dos cenários descritos e recuperar o tempo de resposta de cada invocação. Em um cenário de testes para o X-Road, no *path* e no HTTP *Header* há o endereço de dois *Security Server* nos quais o serviço de teste será invocado durante as requisições. Por outro lado, há um segundo cenário com as mesmas especificações mencionadas acima para duas *proxy* tendo em vista o Kong. Vale ressaltar que nesses experimentos foram utilizadas máquinas virtuais e executados via *command-line interface* (CLI), sendo cada uma com configurações idênticas, como mencionado no tópico anterior, para manter a veracidade entre os resultados.

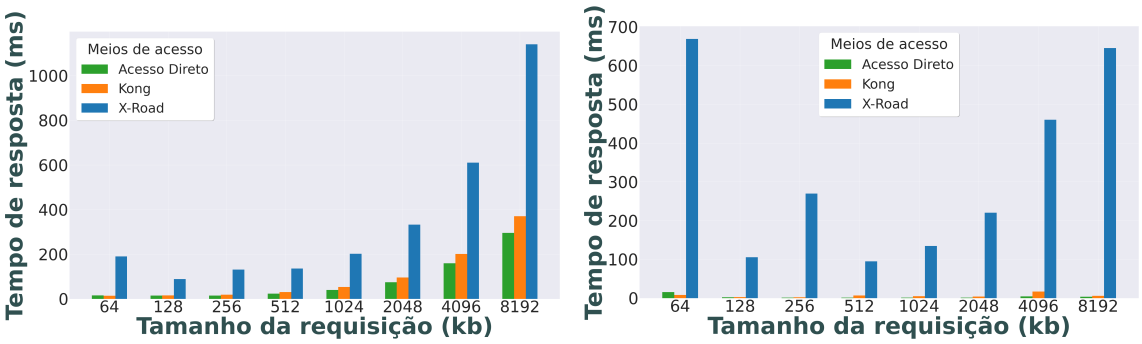
Nesse sentido, foram enviadas mensagens com tamanho de 64 *kilobytes* e duplicada essa grandeza até o valor de 8192 *kilobytes*. Foram realizadas 100 iterações com tempo de inicialização de 1 segundo por usuário para cada tamanho de requisição nos três cenários possíveis: invocação direta, *X-Road* e *Kong*. O versão do gerador de carga é Apache JMeter 5.4.3 .

3.3. Resultados

A Figura 5 apresenta o resultado da classe serial de experimentos. Esse cenário busca mostrar a sobrecarga da solução de interoperabilidade no caso mais básico, sem con-

[4] https://jmeter.apache.org/

corrência. Pelo tempo médio da resposta de uma requisição (Figura 5a) e desvio padrão (Figura 5b), observa-se que a solução baseada em *Kong* apresenta sobrecarga baixa, enquanto o *X-Road* possui um desempenho inferior na escala de ordens de grandeza.



(a) Tempo Médio (b) Desvio Padrão

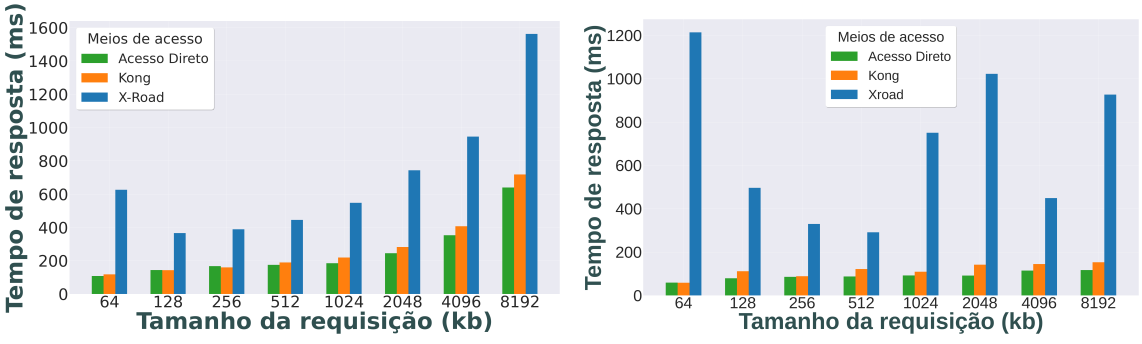
Figura 5. Resultado para a Classe Serial

Considerando os altos tempos de resposta para o *X-Road*, levanta-se a hipótese de que o mesmo poderia lidar melhor com chamadas concorrentes, que é o cenário mais realista, sendo que o tempo serial estaria contabilizando a criação de *threads* e estruturas que só trazem benefício para invocações concorrentes.

A Figura 6 mostra o resultado das invocações concorrentes. Apesar de alguma variabilidade no desvio padrão (Figura 6b), o tempo médio de cada resposta do *X-Road* (6a) continua ordens de grandeza maior do que a invocação direta ou através do *Kong*. Todas as opções exigem um intervalo maior para tratar as requisições. Por exemplo, enquanto na Figura 5a o valor para o tamanho de 4096 KB fica na casa de 600 ms para o *X-Road*, o mesmo tamanho exige 1000 ms na Figura 6a. As máquinas virtuais possuem dois núcleos, sendo que as quatro invocações simultâneas foram usadas para avaliar como cada solução consegue gerenciar *threads* em cenário de recursos limitadas. Os resultados não permitem concluir que qualquer uma das soluções tenha tratamento de concorrência mais eficiente.

4. Conclusão e Trabalhos Futuros

A partir das análises realizadas no contexto deste trabalho, conclui-se que na instalação padrão do *X-Road* e utilizando a arquitetura proposta para interoperabilidade com o *Kong*,



(a) Tempo Médio (b) Desvio Padrão

Figura 6. Resultado para a Classe Paralela

a última opção tem melhor desempenho na invocação de serviços, oferecendo sobrecarga baixa em relação à invocação direta dos serviços. Foi levantada a hipótese do desempenho se equiparar invocações concorrentes, o que não foi o caso.

Como trabalhos futuros pretende-se explicar o desempenho das duas soluções. Primeiro, nesta primeira leva de experimentos, a configuração de *hardware* das máquinas virtuais foi fixa. Planejamos refazer os experimentos, alterando a configuração de memória e número de núcleos para definir qual desses fatores tem maior impacto no desempenho das soluções. O objetivo é avaliar se a partir de determinada configuração de máquina o *X-Road* tem desempenho equiparável com a solução baseada em *Kong*. Segundo, o *X-Road* faz o *time-stamping* de lotes de mensagens, além de armazenar vários registros sobre o estado do funcionamento do *Security Servers*. Apesar da importância desses registros, sua geração e armazenamento tem impacto no desempenho. Pretende-se desabilitá-los e realizar nova análise para medir esse impacto.

Um terceiro trabalho futuro seria aprimorar a arquitetura proposta com o *Kong* para considerar a identidade dos usuários na requisição. No modelo atual, há um canal seguro estabelecido, entretanto não há verificação de credenciais de maneira similar ao *X-Road*. A adição de um serviço SSO (*Single Sign On*) irá aproximar as duas soluções em termos de funcionalidades e terá impacto no desempenho do *Kong*, que também será avaliado em nova análise.

Agradecimentos

Este trabalho é parcialmente financiado pelos projetos FUNCAP 04772314/2020, 04772420/2020 e 04772551/2020.

Referências

Almeida, K. and Costa, R. (2021). Uma proposta de barramento de dados para integração de serviços públicos digitais. In *Anais do IX Workshop de Computação Aplicada em Governo Eletrônico*, pages 25–36, Porto Alegre, RS, Brasil. SBC.

Anthes, G. (2015). Estonia: A model for e-government. *Commun. ACM*, 58(6):18–20.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86–93.

Gannon, D., Barga, R., and Sundaresan, N. (2017). Cloud-native applications. *IEEE Cloud Computing*, 4(5):16–21.

Ministério do Planejamento, O. e. G. (2012). Padrões de interoperabilidade de governo eletrônico. `https://www.gov.br/governodigital/pt-br/governanca-de-dados/Guia_de_Interoperabilidade_Manual_do_Gestor_2012.pdf`. [Última visita online: 2022-03-16].

NIIS, N. I. f. I. S. (2020). X-Road Fundamentals. https://x-road.thinkific.com/courses/x-road-fundamentals. [Última visita online: 2022-03-23].

NIIS, Nordic Institute for Interoperability Solution (2020). X-Road Case Studies. https://x-road.global/piloting-digital-prescriptions-in-germany-through-secure-data-exchange. [Última visita online: 2022-04-06].

Pautasso, C. (2014). Restful web services: principles, patterns, emerging technologies. In *Web Services Foundations*, pages 31–51. Springer.

Raj, G. S. (1998). A detailed comparison of corba, dcom and java/rmi. *Object Management Group (OMG) whitepaper*.

Ramakani, A. (2020). Kong Api Gateway - From Zero to Production. https://medium.com/swlh/kong-api-gateway-zero-to-production-5b8431495ee. [Última visita online: 2022-04-06].

Richardson, C. (2019). *Microservices patterns: with examples in Java*. Manning Publications.