MFog: Uma Arquitetura Resiliente em Névoa para Aplicações do Centro de Operações de Resposta a Desastres

Marcos Francisco da Silva[1], **Aleteia Araujo**[1]

[1]Departamento de Ciência da Computação, Universidade de Brasília – UnB

marcos.silva@aluno.unb.br, aleteia@unb.br

Abstract. *In disaster response operations it is necessary to use tactical and operational software. However, operations can take place in adverse locations with scarce Internet access resources. This limitation can cause problems in the services operation, since these applications are in the cloud. In this context, we propose an architecture that employ the concepts of fog computing, called MFog, focusing on the resilience of applications. A case study was carried out in a simulated environment, and the results showed the feasibility of the model to be applied in a disaster response Operations Center.*

Resumo. *Em operações de resposta a desastres é necessário o uso de softwares táticos e operacionais. No entanto, as operações podem ocorrer em locais adversos com recursos escassos de acesso à Internet. Essa limitação pode gerar problemas no fornecimento de serviços, uma vez que estas aplicações estão na nuvem. Neste contexto, propõe-se uma arquitetura que explora os conceitos de computação em névoa denominada MFog, com foco na resiliência das aplicações. Um estudo de caso foi realizado em ambiente simulado e os resultados evidenciaram a viabilidade do modelo quando aplicado em Centro de Operações de resposta a desastres.*

1. Introdução

Operações de resposta a grande catástrofes são deflagradas para atender ocorrências de furacões, enchentes e outros desastres naturais. Para coordenar essas operações, o Exército Brasileiro (EB) concebeu o projeto Centro de Coordenação de Operações Móvel (CCOp Mv) [1], o CCOp Mv provê aos agentes e militares envolvidos na operação, através de viaturas equipadas com diversos equipamentos, acesso a recursos de Tecnologia de Informação e Comunicação, tais como aplicações de mensagem, email, navegação, comando e controle, videoconferência, etc. Entretanto, essas aplicações estão alocadas na nuvem, sendo o acesso possibilitado através de *link* com a Internet. Nesses locais a conexão com a Internet pode ser limitada e dessa forma, mesmo com a presença das viaturas próximas às operações, os serviços podem não funcionar corretamente, gerando lentidão ou indisponibilidade.

Uma solução para esse problema, é a transferência de parte do processamento da aplicação para próximo do usuário final, fornecendo assim maior resiliência durante os períodos de indisponibilidade de conexão. Os dispositivos, próximos do usuário, formam

[1]http://www.epex.eb.mil.br/index.php/proteger

a camada de névoa [Habibi et al. 2020]. Esse cenário pode ser dividido em três camadas: (i) camada de nuvem com maior capacidade de computação; (ii) camada de névoa, situada próximo ao cliente final, com capacidade de processar, transmitir e armazenar temporariamente os dados recebidos; e (iii) camada de borda, onde estão os dispositivos dos usuários[Habibi et al. 2020].

A maior resiliência dos sistemas podem ser alcançada através de migrações de dados, de processamento e até mesmo das aplicações para nós da névoa, Prokhorenko *et al.* [Prokhorenko and Babar 2020]. Diferentes arquiteturas em névoa foram propostas com uso de técnicas de migração de componentes de aplicação. Na maior parte desses estudos [Santos et al. 2019, Rosário et al. 2018], a solução proposta possibilita a distribuição dos componentes da aplicação pelos nós da névoa e também pela nuvem, de tal forma que os recursos de hardware são utilizados de maneira mais eficiente, fornecendo serviços com melhor desempenho. No entanto, nesses trabalhos não são tratados problemas relativos a desconexão quando um nó da névoa fica isolado dos demais.

Uma pesquisa abrangente sobre diversos modelos arquiteturais foi realizada por Habibi *et al.* [Habibi et al. 2020], na qual os autores apresentam arquiteturas de referência para computação em névoa. Dentre elas, é apresentada a arquitetura SORTS [Velasquez et al. 2017], que possui 3 camadas e pode ser implantada na estrutura do CCOP Mv. Além disso, são propostos os elementos que devem compor o orquestrador, a arquitetura permite a migração e orquestração das aplicações ao longo do conjunto do nós e nuvem. No entanto, não são tratados mecanismos de resiliência. Dessa forma, a principal contribuição deste trabalho é apresentar uma arquitetura resiliente que adota mecanismos que garantem o fornecimento do serviço mesmo em situações de perda de conexão entre os nós.

2. MFog: Arquitetura para Aplicações do CCop Mv

Com base na arquitetura de referência SORTS [Velasquez et al. 2017], foi proposta uma nova arquitetura, com elementos adicionais cujo objetivo é garantir a resiliência de aplicações no caso de desconexão de nó, essa arquitetura esta apresentada na Figura 1.

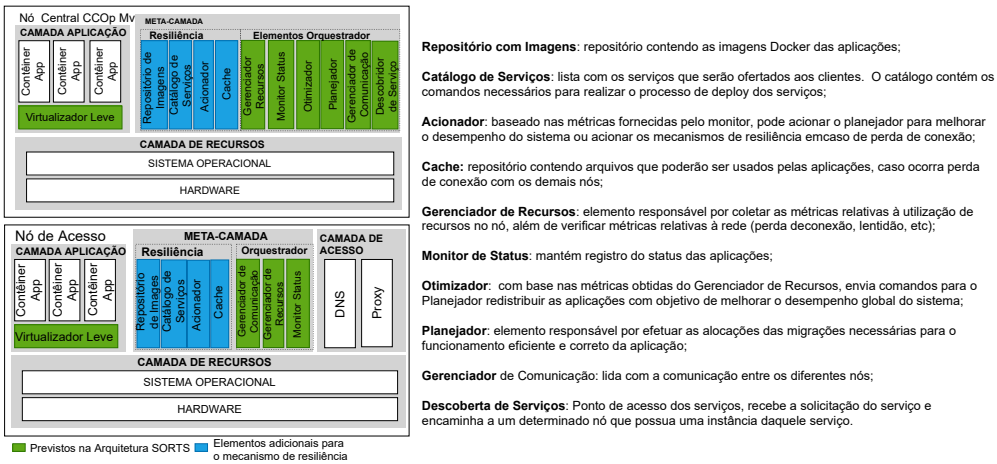

Figura 1. Modelo proposto para arquitetura *MFOG*.

A Figura 2a apresenta cenários de aplicação dessa arquitetura no CCOp Mv (com-

posto por duas viaturas de nó de acesso e uma viatura de nó central, todas possuem recursos computacionais). No exemplo da Figura, uma aplicação é composta por 4 componentes conteinerizados (C_1 à C_4), cada componente faz uma determinada função na aplicação e C_5 é um repositório de arquivos, não podendo ser conteinerizado. Em um cenário, no qual o link com a Internet possui capacidade e está estável, os usuários acessam a aplicação diretamente na nuvem (as viaturas apenas fornecem o acesso). Em um cenário com uso da arquitetura *MFOG*, as aplicações são migradas para a névoa, Figura 2b e os usuários passam a utilizar a aplicação por meio dos caminhões próximos, nesse caso os componentes são distribuídos pelos nós de névoa, mas o acesso dos clientes ocorre de forma transparente. Em um cenário de falha (Figura 2c), um dos nós perde a conexão com o nó central, isso implica em uma situação que nenhum dos clientes conectados a esse nó terá acesso ao serviço fornecido pela aplicação.

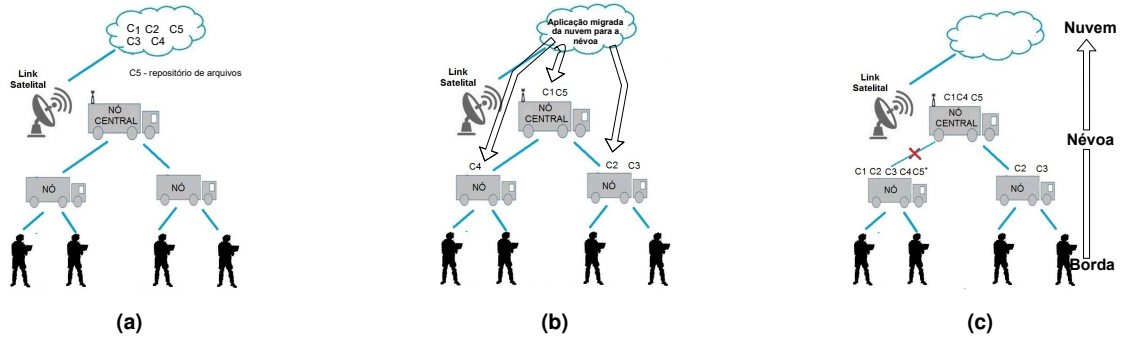

Figura 2. a) aplicações na nuvem, b) na névoa, c) distribuição por falha.

O mecanismo de recuperação da *MFog* atua para instanciar a aplicação no nó desconectado e reestabelecer seu funcionamento para os clientes desse ponto de acesso. O mecanismo possibilita a ativação de componentes da aplicação conteinerizados, no entanto, componentes de armazenamento de arquivos persistentes (C_5 do exemplo anterior) não podem ser migrados (devido ao tamanho do repositório). Nesse caso, a *MFog* passa a utilizar o conteúdo do *cache* contendo parte do componente de armazenamento, formado por arquivos acessados mais recentemente (no exemplo da Figura 2c, o componente passa a ser denominado $C_5{}^*$). Por fim, ocorre uma redistribuição nos outros nós não afetadas da rede, de forma que o sistema também continue em funcionamento para os demais usuários (no exemplo, da Figura 2c, o componente C_4 é migrado para o nó central, pois o mesmo estava alocado no nó desconectado). Ao retornar a conexão com o nó central, o mecanismo remove a implantação realizada e faz o nó de acesso ser reintegrado à estrutura daquele nó, sendo orquestrado por ele.

3. Testes com a Arquitetura Proposta

Para verificar o funcionamento da arquitetura *MFog*, um estudo de caso foi realizado em ambiente simulado. A aplicação em estudo foi a *PicApport* que é um servidor de fotos que permite visualizar e gerenciar fotos em todos os dispositivos conectados à rede, tais como *desktops*, *notebooks*, *tablets* e *smartphones*. Ela possui dois componentes, sendo o componente 1 (C_1) o *front-end* da aplicação, e o componente 2 (C_2) o repositório de fotos. A arquitetura foi construída usando aplicações *open-source*. Assim, o virtualizador escolhido foi o *Docker Engine*. Para a simular a camada de acesso, foi utilizada a aplicação

Bind DNS em conjunto com o *Proxy Squid*. Para simular a meta-camada foi criado um *script* em *python* que faz o papel de Monitor de Status, Acionador e também alimenta o *Cache* com arquivos consumidos pela aplicação. A meta-camada foi implementada de forma mais completa apenas no nó de acesso, com o objetivo de testar o mecanismo de resiliência. A Camada de Recursos foi implantada usando máquinas virtuais sobre o virtualizador *Oracle Virtual Box*. O ambiente foi montado usando um computador pessoal (8 Gb RAM, 1 TB de HD, 4 CPU e *Windows* 10) com uso de máquinas virtuais. Na montagem, as máquinas virtuais simulam os nós e o *Apache Jmeter* simula os clientes realizando requisições. A nuvem foi simulada em uma máquina virtual com os mesmos recursos do nó central, com restrição na placa de rede para o valor de 100 Kbps (criada com a aplicação *wondershaper*[2], simulando um link satelital limitado). Por fim, foi criado um *script* que monitora o conteúdo acessado pelos clientes através do *squid*, e faz cópias dos últimos arquivos acessados no nó central para o *cache* do nó de acesso (simulando a migração do repositório persistente dento do *MFOG*). Para simular o acesso dos clientes com o *JMeter*, foi configurado na ferramenta o valor de 20 requisições por segundo, esse valor foi obtido de forma prévia e cria uma carga suficiente para avaliar o sistema. O tempo de espera (TE) foi definido com o valor de 10s (para que a simulação pudesse ocorrer dentro do intervalo de 1 minuto), a própria ferramenta *JMeter*[3] fornece o tempo de resposta de cada solicitação.

Assim, o experimento foi realizado por meio dos seguintes passos para a comparação de desempenho entre nuvem e névoa: (i) C_1 e C_2 foram alocadas na nuvem; (ii) O *DNS* foi configurado para a URL da aplicação apontar para a nuvem e a simulação é executada por um período de 3 minutos; (iii) C_1 e C_2 foram alocados na névoa (nó central); (iv) O *DNS* foi configurado para a URL da aplicação apontar para a névoa e o passo (ii) foi executado novamente. Para a avaliação do mecanismo de resiliência, foram seguidos os seguintes passos: (i) O *DNS* foi configurado para a URL da aplicação apontar para o nó central; (ii)A simulação é iniciada e após transcorridos 30 segundos, o *link* entre os nós é desconectado, simulando perda de conexão.

4. Resultados

A comparação do desempenho da aplicação quando alocada na nuvem e na névoa pode ser visualizada na Figura 3a. Nessa figura é possível observar que quando a aplicação está alocada na nuvem Figura 3aa, parte das requisições tem tempo de resposta superior a 500 ms e existe a ocorrência de alguns erros. É importante ressaltar que no cenário de emprego do CCOP Mv o *enlace* foi limitado a 100 Kbps . Para o cenário onde a aplicação esta alocada na névoa (Figura 3ab), todas as requisições têm reposta inferior a 500 ms, o que indica um melhor desempenho da aplicação nesse caso. O comportamento da *MFog* durante perda de conexão pode ser visualizado na Figura 3b. Assim, é possível observar que a aplicação funciona no período do início da simulação até a perda de conexão (que ocorre por volta do trigésimo segundo). Após isso, o algoritmo de resiliência aguarda o período TE até tomar a ação de fazer o *deploy* da aplicação no nó de acesso. O processo completo leva cerca de 50 segundos até que a aplicação volte a ficar disponível.

O tempo de 50 segundos, até o serviço ser reestabelecido, é necessário para que ocorra a detecção na perda de conexão (10 segundos), acrescido do tempo para a

[2]`http://manpages.ubuntu.com/manpages/trusty/man8/wondershaper.8.html`
[3]`https://jmeter.apache.org/`

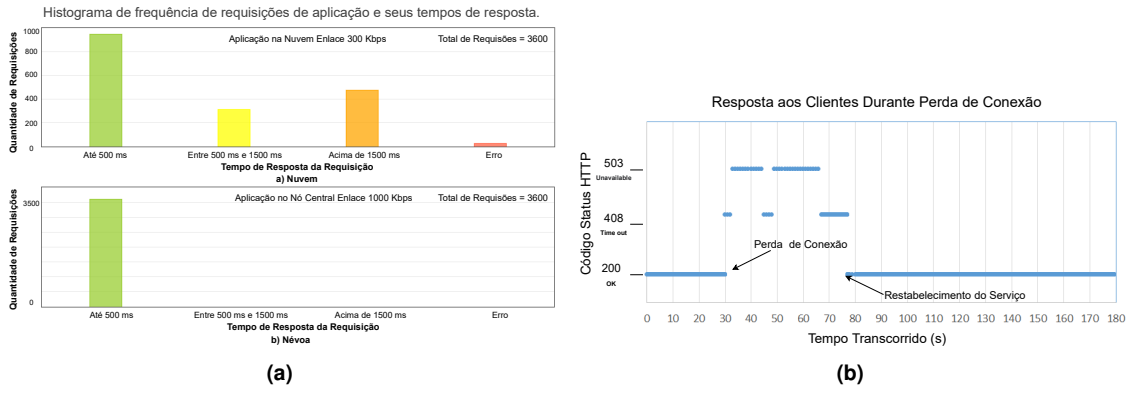

Figura 3. (a) comparação nuvem x névoa, (b) mecanismo de resiliência.

inicialização dos contêineres e da mudança no DNS, ele pode ser reduzido caso o nó possua mais recursos de hardware e processamento. No entanto, para aplicação avaliada, o tempo total de 50 segundos, não implicaria em problemas para o usuário, uma vez que a aplicação fornece mapas e o tempo de indisponibilidade seria menor que 1 minuto.

5. Conclusão

Este artigo apresentou a *MFOG*, Os resultados dos testes preliminares indicaram um baixo desempenho em relação ao tempo de resposta da aplicação alocada na nuvem (quando o *link* tem baixa capacidade). Ademais, os resultados mostraram que um mecanismo de resiliência contra problemas de conexão, pode fornecer uma garantia de funcionamento da aplicação mesmo diante de problemas de conexão por este nó. No entanto, é necessária a realização de estudos mais aprofundados com relação ao tempo de recuperação e técnicas de persistência de *cache*. Em trabalhos futuros serão realizados testes completos com métricas de consumo de hardware e energia, além de cargas que simulam o ambiente real, onde será possível comparar a eficiência do modelo proposto em relação a outros mecanismos de resiliência.

Referências

Habibi, P., Farhoudi, M., Kazemian, S., Khorsandi, S., and Leon-Garcia, A. (2020). Fog computing: a comprehensive architectural survey. *IEEE Access*, 8:69105–69133.

Prokhorenko, V. and Babar, M. A. (2020). Architectural resilience in cloud, fog and edge systems: A survey. *IEEE Access*, 8:28078–28095.

Rosário, D., Schimuneck, M., Camargo, J., Nobre, J., Both, C., Rochol, J., and Gerla, M. (2018). Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support. *Sensors*, 18(2):329.

Santos, J., Wauters, T., Volckaert, B., and De Turck, F. (2019). Towards network-aware resource provisioning in kubernetes for fog computing applications. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 351–359. IEEE.

Velasquez, K., Abreu, D. P., Goncalves, D., Bittencourt, L., Curado, M., Monteiro, E., and Madeira, E. (2017). Service orchestration in fog environments. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 329–336. IEEE.