CloudGI, Gerenciando Instâncias de um Serviço Replicado em uma Plataforma de Computação em Nuvem

Poliana Santos Nascimento, Allan Edgard Silva Freitas

[1]Instituto Federal da Bahia (IFBA), Campus de Salvador, Bahia, Brasil

polianasantos@ifba.edu.br, allan@ifba.edu.br

Abstract. *Nowadays, cloud computing platforms have been used to provide computing services, allowing on-demand rent of computational resources. However, even cloud computing-based services may fail, and replication mechanisms shoud be used, providing an adequate number of replicas to run the service. In this work, we present CloudGI , a replica manager for OpenStack cloud computing platform. Such tool allows for deploy a replicated service, instantianting N replicas for tolerate F faults: crash-recovery or bizantine – according to a prescribed service level. CloudGI monitors replicas, indicate faults and act under fault replicas for fixing them. Also, it implements periodic replica rejuvenation, in order to pro-actively minimizing the number of possible compromised replicas.*

Resumo. *Plataformas de computação em nuvem têm sido amplamente utilizadas para o provimento de serviços computacionais nos dias atuais, permitindo o dimensionamento de recursos sob demanda e a escalabilidade adequada no uso de serviços. Contudo, mesmo serviços providos por meio de um ambiente de Computação em Nuvem estão sujeitos a falhas, e mecanismos usuais de replicação podem ser utilizados, instanciando um conjunto de réplicas adequado para a manutenção do serviço. Neste artigo, apresentamos a ferramenta CloudGI, um gerenciador de réplicas para o ambiente de computação em nuvem OpenStack. Por meio desta ferramenta, pode-se implementar um serviço replicado, instanciando N réplicas para tolerar F falhas – crash-recovery ou bizantinas, de acordo com o nível de serviço estabelecido. CloudGI monitora o funcionamento das réplicas, indica falhas e atua sob as réplicas falhas para re-estabelecer o funcionamento. Ainda, implementa o rejuvenescimento periódico de réplicas sujeitas a falhas bizantinas, de modo a minimizar o número de réplicas comprometidas.*

1. Introdução

Nos dias atuais, a computação em nuvem emerge como o paradigma adequado para hospedagem de infra-estrutura e serviços na Internet, com mecanismos para provisionamento dinâmico e escalabilidade no uso de recursos, conforme a demanda [Zhang et al. 2010].

Há diferentes modos de provimento de serviços da computação em nuvens: (i) *Infrastructure as a Service (IaaS)*, com o provisionamento de recursos de infra-estrutura sob demanda, normalmente na forma de máquinas virtuais; (ii) *Platform as a Service (PaaS)*, com uma plataforma de execução, normalmente na forma de um sistema operacional combinado com um *framework* de desenvolvimento de aplicações; e (iii) *Software as a Service (SaaS)*, com aplicações sob demanda na Internet.

O uso de IaaS para a computação em nuvem permite às organizações a vantagem de manter máquinas virtuais que combinem diferentes plataformas de execução e serviços, nem sempre disponíveis sob a forma de PaaS ou de SaaS, bem como gerenciar de forma mais suave e flexível a alocação de recursos para cada máquina do que com a alocação de máquinas físicas. Isto pode ser obtido por meio da contratação de recursos em uma nuvem pública, como Amazon EC2 [Amazon 2015], ou ainda, através de uma infraestrutura própria ou compartilhada entre um número reduzido de organizações, em nuvens privadas, as quais podem ser construídas a partir do uso de plataformas computacionais como Eucalyptus [Nurmi et al. 2009], OpenNebula [Fontán et al. 2008] ou OpenStack [Sefraoui et al. 2012].

Ainda assim, embora o provimento de recursos computacionais por meio de uma nuvem IaaS facilite o gerenciamento de recursos e a escalabilidade, mesmo sistemas providos sob um ambiente de computação em nuvem estão sujeitos a falhas.

Diferentes mecanismos podem prover serviços em um ambiente computacional sujeito a falhas, como, por exemplo, a replicação ativa [Schneider 1990]. Na replicação ativa, um conjunto de réplicas mantém o mesmo estado do serviço – cada réplica executar exatamente a mesma sequência de operações de clientes, em um comportamento determinístico: dada uma mesma operação e um mesmo estado inicial, cada réplica atualizará seu estado para o mesmo estado final. Embora a replicação ativa possa ser adequada em um cenário de falhas *crash-recovery*, em que réplicas possam falhar e ao retornar sincronizar com o estado atual do conjunto de réplicas, na presença de falhas bizantinas [Lamport et al. 1982], o comportamento de uma réplica comprometida pode não ser determinístico, e cada ação deve ser validada por um quórum superior a dois terços das réplicas (assume-se que menos de um terço das réplicas possa ser comprometido), bem como canais de comunicação podem ser comprometidos, e mecanismos baseados em chaves criptográficas podem ser utilizados para garantir a integridade da comunicação entre duas réplicas não comprometidas. Protocolos como o *Pratical Byzantine Fault-Tolerance* (PBFT) [Castro and Liskov 1999] utilizam destes mecanismos para provimento da replicação bizantina, ou seja, de um serviço de replicação tolerante a falhas bizantinas.

Neste artigo apresentamos a ferramenta CloudGI, um gerenciador de réplicas para o ambiente de computação em nuvem *OpenStack*. Por meio desta ferramenta, pode-se instanciar o número adequado de réplicas de um serviço de modo a tolerar um dado número de réplicas sujeitas a falhas *crash-recovery* ou sujeitas a falhas bizantinas. O número de réplicas falhas toleradas define o nível de serviço provido pela ferramenta ao serviço replicado. CloudGI permite ainda monitorar o funcionamento das réplicas, indicando réplicas falhas, atuar sob réplicas falhas para tentar re-estabelecer o funcionamento das mesmas e rejuvenescer réplicas sujeitas a falhas bizantinas, minimizando, de forma pró-ativa, as réplicas comprometidas.

O restante deste artigo é organizado como segue. A Seção 2 discute os trabalhos correlatos à gerência de réplicas em ambientes de computação em nuvem. A ferramenta proposta para gerência de réplicas para um ambiente de computação em nuvem é proposta na Seção 3, sendo apresentados detalhes de sua arquitetura e implementação na Seção 4. Detalhes da documentação e demonstração de uso são expostos na Seção 5. Finalmente, apresentamos nossas considerações finais na Seção 6.

2. Trabalhos Correlatos

Um *framework* para gerenciamento de recursos de um ambiente de nuvem computacional baseado em *Eucalyptus* [Nurmi et al. 2009] é proposto em [Fraga et al. 2013], denominado: *Just-in-Time (JiT) Clouds*. O *JiT Clouds* permite alocação de recursos de nuvens federadas por meio de diversos *JiT Data Centers*, composto por diferentes módulos, dentre os quais o *JiT DC Middleware*, responsável por gerenciar recursos de máquinas virtuais instanciadas no ambiente do *Eucalyptus*. Outro módulo, denominado *MDA*, oferece um serviço de replicação *primary-backup* para máquinas virtuais com mecanismo de gestão autonômica.

O *Fault Tolerance Manager* (FTM) é um *framework* para gestão de tolerância a falhas em ambientes de computação em nuvem [Jhawar et al. 2012]. O componente *Replication Manager* fornece um serviço de tolerância a falhas que provê replicação de aplicação de usuários, por meio de um mecanismo de replicação *primary-backup*.

Claudia é um sistema de gerenciamento de uma nuvem federada, baseada em diferentes provedores de nuvem [Rodero-Merino et al. 2010]. Este sistema de gerenciamento provê uma camada de abstração única, que ameniza problemas de *lock-in* e permite a federação transparente de nuvens para a execução dos serviços, preocupando-se com o ciclo de vida dos serviços, e permitindo que réplicas de um serviço estejam hospedadas nas nuvens de diferentes provedores, como, por exemplo, em uma nuvem baseada em *OpenStack* e outra baseada em *Eucalyptus*.

TrustWorthy apresenta um sistema de monitoramento de réplicas IaaS baseado no paradigma *Publish-Subscribe*, com suporte a falhas por *crash* e bizantinas, com alarmes ao administrador, mas não apresenta mecanismos automáticos de atuação na plataforma IaaS quando isso for possível [Padhy et al. 2011]. A proposta do CloudGI fornece o monitoramento ao serviço de replicação ativa, com alguns mecanismos de atuação na plataforma de nuvem.

3. CloudGI: Gerenciador de Instâncias de réplicas em Ambiente de Computação em Nuvem

CloudGI permite instanciar máquinas virtuais em uma plataforma de computação em nuvem IaaS correspondentes à réplicas de um serviço replicado. Cada máquina virtual ao iniciar executa a aplicação que possui uma camada de replicação ativa associada. Desta forma, a aplicação se une ao grupo de réplicas e sincroniza o estado com as mesmas.

Diferentes *middlewares* ou servidores de aplicação podem ser utilizados para desenvolver aplicações com esse suporte, por exemplo, pode-se utilizar o GroupPac, uma implementação em código aberto da especificação de tolerância a falhas do CORBA, para desenvolver o suporte da aplicação à replicação ativa, ou ainda, implementar a aplicação sob o servidor de aplicação JOnAS, em um arranjo de replicação ativa baseado no protocolo *Cluster Method Invocation* (CMI). Desta forma, a proposta do CloudGI possui como premissa que as máquinas virtuais instanciadas como réplicas do serviço possuem todo o suporte de software necessário para interação entre si através de técnicas de replicação ativa.

Ainda assim, CloudGI monitora e gerencia as réplicas, atuando na plataforma de computação em nuvem em caso de falhas, podendo reiniciar instâncias, ou em caso de um

estado mais severo de falha, recriar instâncias. Este suporte em um ambiente de replicação ativa permite suportar o modelo de falhas *crash-recovery*, em que réplicas falhas, ao reiniciarem, podem sincronizar o estado com as demais e prosseguem na computação.

Se o *middleware* ou servidor de aplicação utilizado provê suporte a replicação de falhas bizantinas, implementando um protocolo como o PBFT [Castro and Liskov 1999]. Ou seja, se, dentre outras coisas: as réplicas utilizam de chaves criptográficas apropriadas para a comunicação entre si, e o progresso da computação depende da concordância de mais de dois terços das réplicas, CloudGI implementa o mecanismo de rejuvenescimento de réplicas, no qual, periodicamente cada réplica é recriada, assumindo que ao longo do tempo uma réplica pode ser comprometida e esta operação reestabelece o código original da réplica. Neste caso, mecanismos adicionais de salvaguarda, como a manutenção das imagens, que instanciam as máquinas virtuais, em dispositivo protegido de escrita, podem ser implementados.

CloudGI oferece um serviço de criação, monitoramento e gestão de instâncias de máquinas virtuais em um ambiente de computação em nuvem, que atuam como réplicas de um serviço. Ao definir o serviço, pode-se configurar qual o modelo de falhas: *crash-recovery* ou bizantino. Ainda, se define o número F de nós falhos que podem ser tolerados, entre 1 a 3, o que define o número de réplicas a ser configurado. Desta forma, denomina-se três níveis de serviço (bronze, prata e ouro), conforme a Tabela 1.

Tabela 1. Níveis de serviço do CloudGI

Tipo de Serviço	Nº de Réplicas Falhas Toleradas	Número de Réplicas	
		Modo *Crash-Recovery*	Modo Bizantino
Bronze	1	2	4
Prata	2	3	7
Ouro	3	4	10

Em linhas gerais, no Modo *Crash-Recovery*, temos que $N = F + 1$; e no Modo Bizantino, $N = 3F + 1$. Assumimos como premissa que a infra-estrutura de computação em nuvem é configurada sob um ambiente dedicado e com uma reserva adequada de recursos (e.g. rede de computadores, servidores, sistemas operacionais etc.), no qual é possível determinar com alto grau de probabilidade as latências máximas de comunicação e processamento, ou seja, o conjunto de réplicas, os componentes de software do CloudGI, os sistemas operacionais hospedeiros e a plataforma de computação em nuvem utilizada atuarão como um sistema síncrono. Esta premissa pode ser válida apenas para o conjunto de réplicas, uma vez que os clientes do serviço replicado podem estar dispostos sob a Internet. Contudo, tal premissa permite assumir a detecção perfeita de falhas por *crash*.

CloudGI foi desenvolvido para atuar sob a plataforma de computação em nuvem IaaS *OpenStack* [Sefraoui et al. 2012]. Outras plataformas IaaS de código-aberto que poderiam ter sido utilizadas incluem *OpenNebula* [Fontán et al. 2008], *Eucalyptus* [Nurmi et al. 2009] e *Nimbus* [Keahey et al. 2008]. O conjunto de funcionalidades do *OpenStack*, como a compatibilidade com Amazon EC2 e Amazon S3, permitindo a oferta de mesmos serviços da Amazon, mas em uma nuvem privada ou comunitária, bem como uma tendência crescente de usuários em comparativo a outras plataformas de nuvem IaaS [Wen et al. 2012, Sempolinski and Thain 2010] embasou a escolha do *OpenStack*.

3.1. OpenStack

O *OpenStack* é um projeto de código-aberto desenvolvido, inicialmente, pela NASA *(National Aeronautics and Space Administration)* e pela *Rackspace*.

OpenStack é formado por diversos componentes, dentre os quais: NOVA, fornece máquinas virtuais sob demanda para diferentes hipervisores, como, por exemplo, KVM, Xen, VMware ou Hyper-VGlance; NEUTRON, serviço de rede da plataforma, com recursos como de configuração dinâmica ou estática de endereços Internet e de redes virtuais para uso das máquinas virtuais; SWIFT, serviço de armazenamento distribuído; GLANCE, serviço de recuperação e pesquisa de máquinas virtuais; e CINDER, serviço de armazenamento persistente em blocos (volumes) para uso das máquinas virtuais [Rosado and Bernardino 2014].

Provê recursos de elasticidade e escalabilidade horizontal, que permite que todos os serviços e componentes usem a política *shared-nothing*, e possam ser configurados em diferentes servidores.

O *shell script DevStack* instala todos os componentes do *OpenStack* em um único servidor Linux, o que permite uma implantação rápida do ambiente para fins de testes, prototipação e desenvolvimento.

4. Arquitetura e Implementação do CloudGI

Uma visão geral da arquitetura do CloudGI é apresentada na Figura 1.

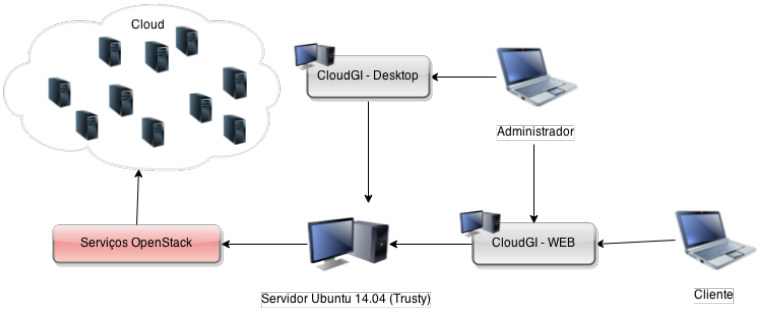

Figura 1. Visão Geral do CloudGI

4.1. Tecnologias Utilizadas

CloudGI é desenvolvido na linguagem JAVA, utilizando a API JAVA Enterprise Edition com servidor de aplicação *Glassfish*. Para persistência dos metadados foi utilizado JDBC e banco de dados *Apache Derby*. O ambiente *OpenStack* foi configurado por meio do script *DevStack* em uma máquina com o sistema operacional *Ubuntu Server 14.04*.

Diante disto, são necessárias algumas especificações para utilização deste serviço. Para o gerenciamento das réplicas foi preciso utilizar os comandos do componente NOVA que permitem dentre outras coisas, criar, excluir e listar instâncias, imagens, *flavors* etc.

4.2. Funcionalidades

CloudGI está sendo desenvolvido em dois módulos: uma aplicação *web* que permite ao cliente configurar as instâncias para a execução do serviço e uma aplicação de gerência

e monitoramento do ambiente. Há dois perfis de usuário, um administrador que possui visão de todos os recursos, e o de cliente que visualiza somente as instâncias associadas aos serviços replicados que configurou.

As principais funcionalidades da aplicação *web* são:

- **Iniciar serviço:** o usuário deve definir o nome do serviço, o nível de serviço e o modelo de falhas associado, a imagem da máquina (que contém a réplica da aplicação instalada) e o *flavor*, uma definição dos recursos da máquina virtual; e
- **Listar instâncias do usuário:** O usuário visualizar todas as instâncias por ele já criadas, incluindo informações sobre modelo de falhas, imagem, nome do serviço, endereço IP da máquina virtual e o estado de execução.

A aplicação de gerência e monitoramento do ambiente é de uso restrito ao perfil administrador e tem como funcionalidades principais:

- **Listar todas as instâncias presentes no servidor:** Visualiza todas as instâncias existentes no ambiente - ver Figura 2;
- **Serviço de monitoramento dos réplicas**: Monitora periodicamente cada instância por meio de um detector de defeitos. Em caso de falha, verifica o estado da instância e intervém para reestabelecer a réplica no estado ativo;
- **Rejuvenesciento de réplicas:** Reinicia periodicamente o estado de réplicas sujeitas a falhas bizantinas;
- **Excluir instâncias manualmente:** Exclui uma instância manualmente;
- **Reiniciar instâncias manualmente:** Reinicia uma instância manualmente; e
- **Iniciar instâncias manualmente:** Cria uma instância manualmente.

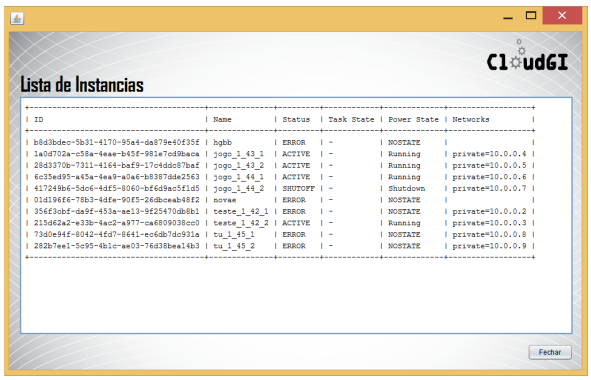

Figura 2. CloudGI - aplicação de Gerência e Monitoramento: Listagem de Réplicas

As máquinas virtuais onde residem as réplicas podem assumir diferentes estados de execução no *OpenStack*, esta informação é utilizada pelo CloudGI para definir a ação mais adequada de recuperação da réplica, conforme a Tabela 2.

5. Documentação e demostração no Salão de Ferramentas do SBRC

No repositório GitHub em https://github.com/PolianaSantos/CloudGI, estão disponíveis: o código-fonte dos módulos do CloudGI (em JAVA) e uma documentação de seu uso. Esta apresenta como implantar uma nuvem de testes *OpenStack* a partir do *DevStack* e instalar e utilizar o CloudGI neste ambiente.

Tabela 2. Estado das instâncias no *OpenStack* e ações possíveis de CloudGI

Status	Descrição	Restauração
ACTIVE	A réplica está ativa	-
BUILD	A réplica está em processo de inicialização	Aguardar a mesma ser iniciada para verificar o estado
ERROR	A réplica falhou em sua criação ou ocorreu algum problema durante a sua execução	Reiniciar a réplica, / (*reboot*) excluir e iniciar uma nova instância com o mesmo nome (*delete e boot*)
PAUSE	Uma réplica foi pausada	Despausar a instância (*unpause*)
SHELVE	Uma réplica esta arquivada	Desarquivar réplica (unshelve)
SUSPEND	Uma réplica foi suspensa	Acordar a maquina (resume)
SHUTOFF	Uma réplica foi desligada	Ligar a maquina (start)

A demonstração da ferramenta apresentará os passos de utilização do CloudGI, os mecanismos de interação da ferramenta com o *OpenStack* e uma execução ao vivo de um serviço replicado com simulação de falhas.

6. Considerações Finais

Neste trabalho apresentamos o CloudGI, um gerenciador de instâncias de réplicas em um ambiente de computação em nuvem IaaS. CloudGI oferece um serviço de criação, monitoramento e gestão de instâncias de máquinas virtuais em um ambiente de computação em nuvem, que atuam como réplicas de um serviço, suportando os modelos de falhas *crash-recovery* ou bizantino e diferentes níveis de serviço, que caracterizam o número F de nós falhos que podem ser tolerados. Esta ferramenta está sendo desenvolvida em JAVA para *OpenStack*, uma das mais utilizadas plataformas IaaS para nuvens privadas ou comunitárias.

Neste artigo apresentamos a ferramenta e suas funcionalidades. CloudGI pode auxiliar na implantação de um serviço replicado, ao possibilitar monitorar as máquinas virtuais em que residem as réplicas da aplicação, bem como interagir com a plataforma de computação em nuvem, recuperando réplicas quando possível.

CloudGI desta forma interage com o usuário, ao configurar o serviço replicado – criando e apresentando o estado das instâncias, e com a plataforma de computação em nuvem *OpenStack*, atuando sob o funcionamento das máquinas virtuais, quando necessário. A ferramenta encontra-se ainda em desenvolvimento. Como funcionalidade futura, uma interface deve permitir às réplicas da aplicação interagir com o CloudGI, o que permite, por exemplo, fornecer informação da camada de computação em nuvem sobre o estado de uma dada réplica para o conjunto de réplicas em execução.

Referências

Amazon (2015). Amazon elastic computing cloud. `http://aws.amazon.com/ec2`.

Castro, M. e Liskov, B. (1999). Practical Byzantine fault tolerance. In *Proc. of the 3rd Symp. on Operating Systems Design and Implementation (OSDI)*, p. 173–186.

Fontán, J., Vázquez, T., Gonzalez, L., Montero, R. S., e Llorente, I. (2008). Opennebula: The open source virtual machine manager for cluster computing. In *Book of Abstracts of Open Source Grid and Cluster Software Conference*.

Fraga, E., Brilhante, J., Costa, R., Brasileiro, F., Bignatto, P., Desani, D., Senger, H., Pereira, A., Garcia, V., Assad, R., et al. (2013). Just-in-time clouds uma abordagem para federação de clouds privadas. In *Anais do Salão de Ferramentas do XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Jhawar, R., Piuri, V., e Santambrogio, M. (2012). A comprehensive conceptual system-level approach to fault tolerance in cloud computing. In *Proc. of IEEE Int. Systems Conference (SysCon)*, p. 1–5.

Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., e Tsugawa, M. (2008). Science clouds: Early experiences in cloud computing for scientific applications. *Cloud computing and applications*, 2008:825–830.

Lamport, L., Shostak, R., e Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., e Zagorodnov, D. (2009). The eucalyptus open-source cloud-computing system. In *Proc. of 9th IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGRID)*, p. 124–131.

Padhy, S., Kreutz, D., Casimiro, A., e Pasin, M. (2011). Trustworthy and resilient monitoring system for cloud infrastructures. In *Proc. of the Workshop on Posters and Demos Track of ACM/IFIP/USENIX 12th Int. Middleware Conf.*, p. 3–4.

Rodero-Merino, L., Vaquero, L. M., Gil, V., Galán, F., Fontán, J., Montero, R. S., e Llorente, I. M. (2010). From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, 26(8):1226–1240.

Rosado, T. e Bernardino, J. (2014). An overview of openstack architecture. In *Proc. of the 18th Int. ACM Database Engineering & Applications Symp. (IDEAS)*, p. 366–367.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Computing Surveys*, 22(4):299–319.

Sefraoui, O., Aissaoui, M., e Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *Int. Journal of Computer Applications*, 55(3):38–42.

Sempolinski, P. e Thain, D. (2010). A comparison and critique of eucalyptus, opennebula and nimbus. In *Proc. of the IEEE 2nd Int. Conf. on Cloud Computing Technology and Science (CloudCom)*, p. 417–426.

Wen, X., Gu, G., Li, Q., Gao, Y., e Zhang, X. (2012). Comparison of open-source cloud management platforms: Openstack and opennebula. In *Proc. of 9th Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD)*, p. 2457–2461.

Zhang, Q., Cheng, L., e Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.